ABSTRACT

Title of Document: APPLICATION OF STOCHASTIC RELIABILITY MODELING TO WATERFALL AND FEATURE DRIVEN DEVELOPMENT SOFTWARE DEVELOPMENT LIFECYCLES

David M. Johnson, Master of Science, 2011

Directed By: Dr. Carol Smidts, Advisor
Professor
Department of Mechanical and Aerospace Engineering

There are many techniques for performing software reliability modeling. In the environment of software development some models use the stochastic nature of fault introduction and fault removal to predict reliability. This thesis research analyzes a stochastic approach to software reliability modeling and its performance on two distinct software development lifecycles. The derivation of the model is applied to each lifecycle. Contrasts between the lifecycles are shown.

Actual data collected from industry projects illustrate the performance of the model to the lifecycle. Actual software development fault data is used in select phases of each lifecycle for comparisons with the model predicted fault data. Various enhancements to the model are presented and evaluated, including optimization of the parameters based on partial observations.

APPLICATION OF STOCHASTIC RELIABILITY MODELING TO WATERFALL AND FEATURE DRIVEN DEVELOPMENT SOFTWARE DEVELOPMENT LIFECYCLES

By

David M. Johnson

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2011

Advisory Committee:
Professor Carol Smidts, Advisor
Professor Mohammad Modarres, Chair
Professor Ali Mosleh, Advisor

Dedication

This thesis is dedicated to my wife Ann and sons Patrick and Paul who supported and encouraged me throughout all my years of study. My wife has inspired me to continue on with this effort on many occasions and deserves so much credit for this thesis. My sons, with their scholarly achievements, have given me a desire to attain educational goals just as they have. I hope to have gained experiences in this work to pass on to my sons. My family has provided me with the desire and motivation to persevere and the will to continue even when I felt like there was no end in sight. Their confidence in me was my inspiration.

My family is my reason and inspiration for all I do and my hope is that we all benefit from this work and celebrate the achievement.

I would also like to dedicate this thesis to my parents who have been role models throughout my life.

Acknowledgements

I would like to express an immeasurable amount of gratitude to Dr. Carol Smidts for guiding me through all these years of study. Her guidance and support has been an inspiration to continue through my coursework and research. She has provided me with the tools for continuing research, has promoted my ideas, and opened up many avenues for my career. I feel I have received more of an education than the Master's program provides from my interactions with Dr. Smidts. I would also like to thank her for continuing to mentor me in her career at The Ohio State University after leaving the University of Maryland.

I want to gratefully thank Dr. Mohammed Modarres for agreeing to be my co-advisor at the University of Maryland. He has been an invaluable resource and guide. He has always been encouraging and helpful.

I want to thank the members of my thesis committee, Dr. Carol Smidts, Dr. Mohammed Modarres and Dr. Ali Mosleh, for agreeing to serve on my committee and for comments and ideas.

I would like to thank Dr.Charley Tichenor for supplying his expertise in Function Point Analysis to my research. Dr. Tichenor was available on numerous occasions and was always very insightful and giving. This research could not have happened without Dr. Tichenor's generous gift of knowledge.

I would like to thank Mr. Capers Jones for lending guidance in Software Metrics. Mr. Jones provided answers on several occasions to many questions related to Software Metrics. As an industry acknowledged expert in this area it was invaluable to be able to reach out to and personally contact Mr. Jones.

I would like to thank Dr. Laurie Williams for providing insights into Agile software development. Dr. Williams is a valuable asset with great insights and abilities to teach Agile concepts.

I am grateful to Dr. Ying Shi for all of her knowledgeable advice during my graduate studies. Dr. Shi provided insights into the graduate process along with valuable sources of data. She was a valuable resource and expert guide.

I would like to thank Mr. Matt Gerber for all of his assistance and timeliness. I am grateful for his efforts.

Many thanks go to my co-workers and management for their support. The assistance provided as been very valuable. In listening to research ideas they provided valuable comments.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents v

List of Tables x

List of Figures xiv

List of Abbreviations xvi

Chapter 1: Introduction 1

1.1 Background 1

1.2 Motivation 3

1.3 Research Objective 4

1.4 Literature Review 5

1.4.1 Software Metrics 6

1.4.2 Software Reliability Models 8

1.4.3 Software Development Lifecycles 11

1.4.4 A Stochastic Model of Fault Introduction & Removal During Software Development 15

1.5 Research Contributions 16

Chapter 2: Development of the Reliability Model for Waterfall Development 20

2.1 The Software Reliability Model 20

2.1.1 Non-Homogenous Multiple Birth-Death-With-Immigration Process 21

2.1.2 NHMBDWI Process Model Development for the Waterfall Lifecycle.. 22
2.1.2.1 Model for Development of the Kolmogorov Forward Equations... 22
2.1.3 Estimate of the Fault Introduction Rate 27
2.1.4 Estimate of the Expected Change in Fault Count Due to One Repair 30
2.1.5 Estimate of the Fault Removal Rate for Phase of Origin..... 30
2.1.6 Estimate of the Fault Removal Rate for Removal Phases 33
2.1.7 The Success Liklihood Index..... 36
2.1.8 Criticality of Faults 37
2.2 Waterfall Lifecycle Description..... 38
2.2.1 Waterfall Lifecycle Phases 42
2.2.1.1 Markov Model of the Waterfall Lifecycle 43
2.3 The Waterfall Project..... 55
2.3.1 Waterfall Project Function Point Analysis 55
2.3.2 Waterfall Project Parameter Definitions and Derivations 60
2.3.3 Waterfall Project Data..... 66
2.3.4 Waterfall Project Software Reliability Model Results..... 69
Chapter 3: Development of the Reliability Model for Feature Driven Development (FDD) – An Agile Approach 72
3.1 FDD Lifecycle Description..... 72
3.1.1 FDD Lifecycle Phases..... 76
3.1.2 Markov Model of the FDD Lifecycle 79
3.2 The FDD Project..... 82

3.2.1 FDD Project Function Point Analysis 83
3.2.2 FDD Project Parameter Definitions and Derivations 89
3.2.3 FDD Project Data 95
3.2.4 FDD Project Software Reliability Model Results 100
Chapter 4: Enhancements and Optimizations 103
4.1 Insertion of SLI into Fault Removal Phases 103
4.1.1 Waterfall Project Software Reliability Model Results with Fault Removal SLI 106
4.1.2 FDD Project Software Reliability Model Results with Fault Removal SLI 111
4.2 Optimization of the SLI Parameter 116
4.2.1 Analysis of the Model Equation Solving for the Value of Actual Faults (μ_U Observed) 116
4.2.2 Optimization Approaches 123
4.2.3 Project Data with SLI Optimization 125
4.2.4 Analysis of Predicted to Observed Fault Data 136
4.2.4.1 Waterfall Project 137
4.2.4.2 FDD Project 140
Chapter 5: Software Tools for Model Analysis 144
5.1 Waterfall Project Model Analysis Tool 144
5.2 FDD Process Model Analysis Tool 145
5.3 SLI Optimization Tool 146
Chapter 6: Conclusions and Further Research 148

6.1 Conclusions 148
6.2 Further Research 151
6.2.1 Extending the SLI Optimization to Preceding Phases 151
6.2.2 Extending the Model to Other Lifecycles 152
6.2.3 Hidden Markov Model to Optimize Parameters 152
6.2.4 Bayesian Analysis of Model Parameters 153
6.2.5 Phase Time 153
Appendix A. Software Metric Data 154
Appendix B. Waterfall Project SLI from Expert Opinion 156
B.1 RQ SLI from Expertise 156
B.2 RR SLI from Expertise 157
B.3 DE SLI from Expertise 158
B.4 DR SLI from Expertise 159
B.5 CO SLI from Expertise 160
B.6 ST SLI from Expertise 161
Appendix C. FDD Project SLI from Expert Opinion 162
C.1 RQ SLI from Expertise 162
C.2 RR SLI from Expertise 163
C.3 SD SLI from Expertise 164
C.4 SDR SLI from Expertise 165
C.5 SM SLI from Expertise 166
C.6 SMR SLI from Expertise 167
C.7 FL SLI from Expertise 168

C.8 FLR SLI from Expertise 169
C.9 FLP SLI from Expertise 170
C.10 DE(Iteration 0, 1, 2, 3) SLI from Expertise 171
C.11 DR(Iteration 0, 1, 2, 3) SLI from Expertise 172
C.12 CO(Iteration 0, 1, 2, 3) SLI from Expertise 173
C.13 CI(Iteration 0, 1, 2, 3) SLI from Expertise 174
C.14 ST(Iteration 0, 1, 2, 3) SLI from Expertise 175
Appendix D. Test Case Parameters 176
D.1 Waterfall Project 176
D.1.1 Test Case #40 176
D.1.2 Test Case #41 178
D.1.3 Test Case #39 180
D.2 FDD Project 183
D.2.1 Test Case #27 183
D.2.2 Test Case #28 186
D.2.3 Test Case #29 190
Bibliography 195

List of Tables

Table 1 Historical Categorization of Selected Software Reliability Models 9

Table 2 Influencing Factors 37

Table 3 Criticality Categories 38

Table 4 Waterfall Project – Unadjusted Function Point Table 57

Table 5 General System Characteristics – Waterfall Project 59

Table 6 Defect Potential (DP_{φ} * $fd_{j,\varphi}$) per Function Point per Phase 61

Table 7 Mean Effort per Function Point per Lifecycle Phase φ, in Staff Hours 61

Table 8 Boundary Information for $DP_{\varphi} \bullet fd_{j,\varphi}$ and $\bar{t}_{FP,\varphi}$ 62

Table 9 Boundary Information for $\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi}$ 62

Table 10 Values of F for the Fault Introduction Phases 63

Table 11 Fault Potential per Function Point, DP 64

Table 12 Software Defect Origin Percent by Industry Segment per Phase 65

Table 13 Mean Effort In Staff Hours Per Function Point per Phase ($t_{FP,\varphi}$) 67

Table 14 Estimated Staff Hours per Phase from Function Points 67

Table 15 Lifecycle Phase In Staff Hours (Estimated and Actual) 67

Table 16 Observed & Repaired Faults Found per Phase (Waterfall) 69

Table 17 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle 70

Table 18 Feature Driven Development (FDD) Lifecycle Phase 75

Table 19 FDD Project – Function Point Analysis 87

Table 20 General System Characteristics – FDD Project 88

Table 21 $\bar{t}_{fp,\varphi}$ Mean Effort per FP per Lifecycle Phase φ, in Staff Hours 90

Table 22 Boundary Information for $DP_{\varphi} \bullet fd_{j,\varphi}$ and $\bar{t}_{FP,\varphi}$ 91

Table 23 Values of $F_{j,\varphi}$ for the Fault Introduction Phases 91

Table 24 Summary of the Data Required to Calculate $\{\nu(t) * \mu_H(t)\}_{j,\varphi}$ – FDD Project 94

Table 25 Mean Effort In Staff Hours Per Function Point Per Phase ($t_{FP,\varphi}$) – FDD Project 95

Table 26 Lifecycle Phase In Estimated Staff Hours from Function Points – FDD Project 96

Table 27 Lifecycle Phase In Staff Hours (Estimated and Actual) – FDD Project..... 97

Table 28 Observed & Repaired Faults Found per Phase (FDD) 99

Table 29 Model Expected Observed Faults Vs. Observed Faults – FDD Lifecycle 101

Table 30 Ranges Of Defect Removal Efficiency..... 106

Table 31 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle – SLI in Removal Phases 107

Table 32 SLI Values – Baseline and SLI in Removal Phases Models - Waterfall... 110

Table 33 Model Expected Observed Faults Vs. Observed Faults – FDD Lifecycle – SLI in Removal Phases 112

Table 34 SLI Values – Baseline and SLI in Removal Phases Models - FDD 115

Table 35 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle – SLI Optimized in FI-FR Phase Pairs 127

Table 36 Model Expected Observed Faults Vs. Observed Faults –Waterfall Project – SLI Optimized in FI-FR Phase Pairs (Without ST Phase)..... 127

Table 37 SLI Values – Baseline, SLI in Removal Phases and SLI Optimization Models and Expert Opinion – Waterfall Project 130
Table 38 Model Expected Observed Faults Vs. Observed Faults –FDD Project – SLI Optimized in FI-FR Phase Pairs 131
Table 39 Model Expected Observed Faults Vs. Observed Faults –FDD Project – SLI Optimized in FI-FR Phase Pairs (Without ST Phases) 132
Table 40 SLI Values – Baseline, SLI in Removal Phases and SLI Optimization Models and Expert Opinion – FDD Project 135
Table 41 Analysis of Model Results for Waterfall Project 137
Table 42 Analysis of Model Results for FDD Project 140
Table 43 Model Predicted Fault Count at Lifecycle End – Waterfall and FDD 142
Table 44 Patterns of Defect Prevention and Defect Removal Activities 154
Table 45 Defect Removal Efficiencies by Defect Origin 155
Table 46 RQ SLI Calculation From Expertise (Waterfall) 156
Table 47 RR SLI Calculation From Expertise (Waterfall) 157
Table 48 DE SLI Calculation From Expertise (Waterfall) 158
Table 49 DR SLI Calculation From Expertise (Waterfall) 159
Table 50 CO SLI Calculation From Expertise (Waterfall) 160
Table 51 ST SLI Calculation From Expertise (Waterfall) 161
Table 52 RQ SLI Calculation From Expertise (FDD) 162
Table 53 RR SLI Calculation From Expertise (FDD) 163
Table 54 SD SLI Calculation From Expertise (FDD) 164
Table 55 SDR SLI Calculation From Expertise (FDD) 165

Table 56 SM SLI Calculation From Expertise (FDD) 166
Table 57 SMR SLI Calculation From Expertise (FDD) 167
Table 58 FL SLI Calculation From Expertise (FDD) 168
Table 59 FLR SLI Calculation From Expertise (FDD) 169
Table 60 FLP SLI Calculation From Expertise (FDD) 170
Table 61 DE (0, 1, 2, 3) SLI Calculation From Expertise (FDD) 171
Table 62 DR (0, 1, 2, 3) SLI Calculation from Expertise (FDD) 172
Table 63 CO (0, 1, 2, 3) SLI Calculation From Expertise (FDD) 173
Table 64 CI (0, 1, 2, 3) SLI Calculation From Expertise (FDD) 174
Table 65 ST (0, 1, 2, 3) SLI Calculation From Expertise (FDD) 175

List of Figures

Figure 1 Waterfall Software Development Lifecycle Representation 11

Figure 2 FDD Lifecycle Representation 13

Figure 3 Software Development Best Practices by Application Size 14

Figure 4 Injection and Removal of Software Faults in a Software Development Lifecycle 20

Figure 5 Kolmogorov Forward Equation State Transition Diagram of 24

Figure 6 Waterfall Software Development Lifecycle Phases 42

Figure 7 Waterfall Lifecycle Markov Model 44

Figure 8 Inside the RQ Phase 46

Figure 9 Inside the RR Phase 47

Figure 10 Inside the DE Phase 49

Figure 11 Inside the DR Phase 50

Figure 12 Inside CO Phase 52

Figure 13 Inside CI, UT, IgT, ST and AT Phases 54

Figure 14 Estimated vs. Actual Staff Hours – Waterfall Project 68

Figure 15 Model Predicted Fault Count – Waterfall Lifecycle 70

Figure 16 Feature Driven Software Development Lifecycle Phases (Pre-Iteration) 77

Figure 17 Feature Driven Development Software Lifecycle Phases (Iterations) 78

Figure 18 FDD Lifecycle Markov Model 81

Figure 19 Estimated vs. Actual Staff Hours – FDD Project 98

Figure 20 Model Predicted Fault Count – FDD Lifecycle 101

Figure 21 Model Predicted Fault Count – Waterfall Project – SLI in Removal Phases 107
Figure 22 Comparison of Predicted Number of Faults – Baseline and SLI in Removal Phases Models– Waterfall..... 109
Figure 23 Model Predicted Fault Count – FDD Project – SLI in Removal Phases.. 111
Figure 24 Comparison of Predicted Number of Faults – Baseline and SLI in Removal Phases Models– FDD..... 114
Figure 25 Model Predicted Fault Count – Waterfall Project – SLI Optimized in FI-FR Phase Pairs 126
Figure 26 Comparison of Predicted Number of Faults – Baseline, SLI in Removal Phases and SLI Optimization Models– Waterfall Project 129
Figure 27 Model Predicted Fault Count – FDD Project – SLI Optimized in FI-FR Phase Pairs 130
Figure 28 Comparison of Predicted Number of Faults – Baseline, SLI in Removal Phases and SLI Optimization Models– FDD..... 134
Figure 29 Model Predicted to be Observed Faults vs. Actual Observed Fault Data for Waterfall Project 138
Figure 30 Model Predicted to be Observed Faults vs. Actual Observed Fault Data for for FDD Project..... 141
Figure 31 Waterfall Project Analysis Tool Graphical User Interface..... 144
Figure 32 FDD Project Analysis Tool Graphical User Interface..... 145
Figure 33 SLI Optimization Tool Graphical User Interface 146

List of Abbreviations

ANSI	American National Standards Instititute
AT	Acceptance Test Phase
BWDI	Birth-Death-With-Immigration
CO	Coding Phase
CI	Code Inspection Phase
DE	Design Phase
DET	Data Element Type
DR	Design Review Phase
DRE	Defect Removal Efficiency
EI	External Input
EIF	External Interface File
EO	External Output
EQ	External Inquiries
FDD	Feature Driven Development
FL	Feature List Phase
FLP	Feature List Planning Phase
FLR	Feature List Review Phase
FTR	File Types Referenced
FPA	Function Point Analysis
GSC	General System Characteristics
HEP	Human Error Probability

HMM	Hidden Markov Model
IFPUG	International Function Point Users Group
IgT	Integration Test Phase
ILF	Internal Logic File
NHPP	Non-Homogeneous Poisson Process
NHMBWDI	Non-Homogeneous Multiple Birth-Death-With-Immigration
PIF	Performance Influencing Factors
RET	Record Element Type
RQ	Requirements Phase
RR	Requirements Review Phase
SD	System Design Phase
SDR	System Design Review Phase
SLI	Success Likelihood Index
SLIM	Success Likelihood Index Methodology
SM	System Model Phase
SMR	System Model Review Phase
ST	System Test Phase
TDI	Total Degree of Influence
UT	Unit Test Phase
VAF	Value Adjustment Factor
XP	Extreme Programming

Chapter 1: Introduction

1.1 Background

The process of software development contains elements of stochastic behavior in a defined lifecycle process. The measuring of the reliability of software development and the ability to identify the injection and removal of faults in the various phases of software development motivate research into software reliability and software development. There exist many software reliability models. Some of these incorporate the stochastic elements of software development. There also exist many approaches to developing software. The process of software development is itself a stochastic process. Errors and faults are introduced to and removed from the software package as the development process advances through the lifecycle phases. This research applies a published software reliability model based on stochastic processes to two different software development lifecycles; a waterfall lifecycle and a feature driven development lifecycle of software development.

Software reliability models consider the principal factors that affect software reliability. These factors are fault introduction, fault removal and the environment [1]. These software reliability models can be used for estimation or for prediction. The estimation models use statistical inference procedures [1] in late stages of development; the testing phases, to create reliability models [25]. The prediction models use characteristics of the software itself and the process used to develop the software in the prediction of software reliability [25].

Many software reliability models have been advanced in the literature in all the categories of estimation and prediction [1] [2] [3].

A software development lifecycle has traditionally followed phases organized as in a waterfall sequence. In this lifecycle, development of requirements, design, code and tests are done in phases with a corresponding review phase following these development phases. This approach requires that the developer has the knowledge to develop the requirements before knowledge of the design or code. This is referred to as "up-front" requirements development. In this lifecycle phases are sequentially traversed with each phase completed prior to moving to the next phase. In many cases there are many unknowns prior to development of requirements, design, code and tests which require rework to be done as the system becomes more defined.

To alleviate the rework and improve other aspects of software development the Agile software development lifecycle has been advocated. The proponents of Agile proposed an "Agile Manifesto" in February of 2001 and created the Agile Alliance. The manifesto brought forth by this group states [4]:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." This manifesto has led to the development of agile practices, methods and principles.

The Agile approach takes on many variants. Some of these include lean development [26], extreme development [26], pair programming [26], scrum [26], test-driven development [26] and feature driven development [5] [26]. Feature Driven Development (FDD) is an Agile development methodology formed from the Agile Alliance. FDD is defined as an iterative and incremental software development process [5].

1.2 Motivation

The reliability community has proposed numerous software reliability models. Many of these models are proposed and remain theoretical in nature. A need has been identified to validate the models in industry. The validation requires that both the model and software development process be well understood. Another important aspect of software reliability is to understand the effects of the human reliability component of software development. As a result, the understanding of a software reliability model that incorporates the concepts of software development and the human component of that software development is needed.

Software development has evolved as the discipline of software engineering has matured. The software community has transitioned from a waterfall approach to a less structured approach of software development. At this date, in software companies, developers may choose to use a waterfall, a spiral, iterative or other software development approach. Each of these is somewhat similar in that there is a

design, review and build portion of development at each lifecycle phase. An Agile approach uses an iterative approach with more emphasis on the people developing the software than the documentation developed from the process.

Currently, software reliability models rely on a traditional waterfall type approach to software development. The ability to integrate both traditional waterfall and Agile approaches into a software reliability model will create a means for comparison of software reliability data. Thus, there exists a need for a software reliability model which can be applied to various software development lifecycles, generate fault data and also reliability metrics for each development approach. Also, there is much to be done to validate early prediction models.

1.3 Research Objective

There are a number of advantages to be gained by verifying the results of a software reliability model against varying software development lifecycles. The lifecycles can be compared to gain insights into fault introduction rate, fault removal rate, contributing factors to fault introduction and removal and reproducibility of reliability predictions. Across the array of software development there exist many software development lifecycles. An objective of this research is the desire to compare these various lifecycles to one software reliability model.

The objectives of this research are to link a software reliability model with various software development processes, namely traditional waterfall and the Agile Feature Driven Development processes. The research will apply the model to industry projects and gather actual data to analyze in relation to model data. Based on the comparison of the industry data and model data we have shown improvements to

the model, interpreted the analysis of the data and identified areas of the model which may be of concern in applying the model to industry processes. The need to apply theoretical software reliability models to actual industry data and varying software development lifecycles is the goal of this research.

1.4 Literature Review

Software systems are a part of many modern commercial, industrial, military and consumer products. Jones comments on the fact that computers and software are now the driving force of modern business, government and military operations, but can be troublesome, expensive and error prone [10] [11]. With software being an essential part of many systems and the inclusion of software in many safety critical systems there is a need for research into software reliability. The study of software reliability requires an understanding of the measurements or metrics that can be gathered from a software product and the process used to develop that software product. A major part of the software development process is the lifecycle used to create the software product.

Software metrics have been gathered from industry and research for many years. Software metrics can be defined as the measure of the properties of software, the process used to develop the software or the project in which software has been developed. Metrics are used to measure and predict productivity, quality and reliability. Many in research and industry have made these metrics available to the computing communities.

Software reliability is described by Musa et al [1] as the probability of failure-free operation of a computer program for a specified time in a specified environment.

A software reliability model is used to characterize the software reliability of developed code and the process used to develop the code. Musa et al [1] states that the principal factors affecting software reliability are fault introduction, fault removal and the development environment. Musa et al [1], Neufelder [2] and Xie [3] discuss much of the history and types of software reliability modeling. Neufelder [2] describes the objectives of software reliability modeling as a way to evaluate software quantitatively during development, testing and scheduling. Another aspect of software reliability modeling that Neufelder [2] describes is the ability to monitor changes in reliability during development and operation.

1.4.1 Software Metrics

Measurement provides the information necessary to be successful in all of science, engineering and business. Historically software development has lagged behind other engineering endeavors in establishing software metrics for software development. In many areas of software development, the act of collecting metrics is done to acquire data for sizing, estimating, managing and controlling software projects [6] [7].

Jones [7] suggests that metrics be classified as information in the form of hard data, soft data or normalized data. Hard data refers to items which can be quantified with little to no subjectivity. These include items such as cost, time, quantities and efforts. Soft data concerns measurements in which human evaluations are performed. These include project team skill, schedule pressure, expertise and other subjective observations. Normalized data takes the form of standard metrics used for comparative purposes. These include sizing and productivity measures.

Kan [8] suggests that software metrics are classified into three categories: product metrics, process metrics and project metrics. Product metrics describe the characteristics of the product such as size, complexity and performance among others. Process metrics can be used to improve software development and include metrics such as defect removal efficiency (DRE). Product and process metrics contain hard data and normalized data. Project metrics describe attributes of the project development such as number of programmers and schedule stress. Project metrics contain soft data.

Some examples of software metrics commonly used include bugs per line of code (Gaffney estimate), cohesion, completeness, cyclomatic complexity , complexity measures, defect density , fault number days, function point analysis, mean time to failure, requirements traceability, software capability maturity model and test coverage [23] [27]. There are many more software metrics. Software metrics, such as these, often measure a ratio, proportion, percentage or rate. They all have a specific purpose and may have a different definition for each development phase.

Lines of code (LOC) are used as a method of sizing software. This metric is noted for the ambiguity in its operational definition, which is, the performance of actual counting [8]. Another approach to measure size is the use of function points.

Function Point Analysis is used to generate a function point count. Research suggests that function point analysis is an accepted and well known approach to determining the size and complexity of a software component [6] [7] [8] [9]. This count is a unit-of-work measure. Function point analysis has the ability to measure the size of any software deliverable in logical, user oriented terms [9]. Garmus et al

[9] outlines how function point analysis uses the common components of software systems (external inquiries, external inputs, external outputs, internal logical files and external interface files) to compute a measure of the size of a software development project. The function point size is used for various activities including computing costs per function point, outcome prediction or program monitoring [9]. In this research software metrics and function point analysis are used to assist a software reliability model predict the faults occurring during software development phases.

1.4.2 Software Reliability Models

Musa [1], Neufelder [2], and Xie [3] discuss the characteristics, classifications, types and uses of software reliability models. Models are affected by the main factors of software development: fault introduction, fault removal and the development environment. Models are most often time-based. They are often constructed in terms of random processes because of the probabilistic nature of the factors used in their construction. They are often thought to be a Markov model based on the birth-death process of faults in software development.

Models can either provide an estimate or a prediction of the software reliability of a system. An estimation model applies a statistical process to failure data taken from the software and is used primarily in testing phases. A prediction model determines from the characteristics of the software and the development process a prediction of the reliability of the software and is used prior to coding. The predictive models use empirical project data to predict, even before coding begins, what the reliability will be.

Musa [1] states that software reliability has three uses. The first, is to evaluate software engineering technology quantitatively, second, to evaluate development status during test phases and third to monitor the operational performance of software and to control new features added and design changes made to the software.

The following table classifies various software reliability models [1], [2], [3], [12] [25].

Table 1 Historical Categorization of Selected Software Reliability Models

Project Phase	Type	Category	Sub-Category	Name	Description / Investigator
Early Prediction	Predictive			Rome Air Development Center TR-87-171	Predicts a Fault Density Estimate
				Phase-based Model (Gaffney)	Uses fault statistics obtained during technical reviews of requirements, design and code to predict reliability during test and operation.
				Stochastic Model of Fault Introduction & Removal	Model that relates the software failure intensity function to development and debugging errors throughout software life-cycle phases.

Project Phase	Type	Category	Sub-Category	Name	Description / Investigator
Late Prediction	Monolithic	Reliability Growth	Finite Failure – Poisson Type	Exponential	Musa Basic (1975), Okumoto (1979), Goal - Okumoto
			Finite Failure – Binomial Type	Exponential	Jelinski – Moranda (1972), Shooman (1975), Lipow modified Jelinski – Moranda
				Weibull	Schick – Wolverton (1973)
			Infinite Failure – Poisson Type	Geometric	Musa-Okumoto Logarithmic Poisson Execution Time Model
			Infinite Failure	Other	Littlewood and Verrall (1973)
		Input Domain Models	Testing Success		Nelson (1976, 1978), Bastani / Ramamoorthy (1982), Reliability estimated from test cases and number of failures in test cases.
	Structural Architectural Compound Models	Module Based			
		Functional Based			

1.4.3 Software Development Lifecycles

Software is developed using a process called a lifecycle. Lifecycles impose a structure on the development of software. There have been many types of lifecycles. Prominent among these have been the waterfall, spiral and iterative methodologies. In the 2000's the rise of Agile approaches to software development arose. Included in these Agile development approaches are Extreme Programming (XP) [26], Feature Driven Development (FDD) [5], Crystal Clear [28] and Lean Development [26] among others.

A traditional and prominent approach to software development is the waterfall model. This model is a sequential process in which the flow of development proceeds steadily downward, as in a waterfall. This process usually contains at least the phases of requirements definition and analysis, design, implementation (which consists of coding, unit and integration testing), system and acceptance testing, maintenance and retirement. This lifecycle is illustrated below, see Figure 1.

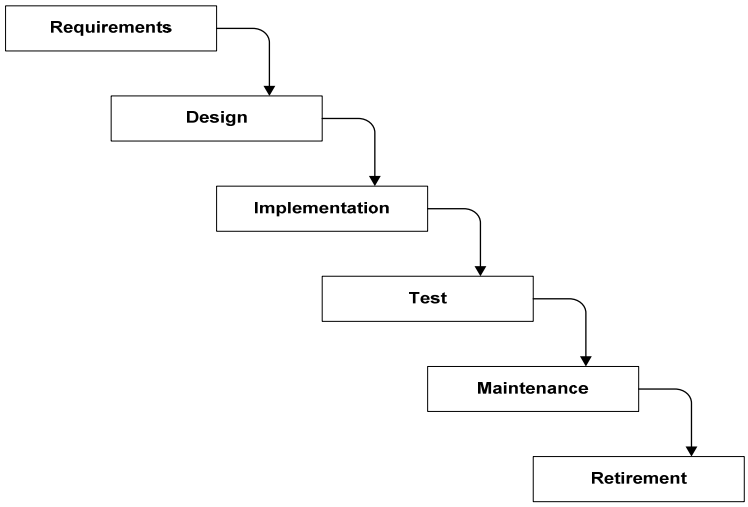


Figure 1 Waterfall Software Development Lifecycle Representation

The waterfall model was introduced in the 1970's by Royce [13]. The premise of the lifecycle is to proceed down the waterfall to a succeeding phase only after completion of the preceding phase. This approach has been followed to varying degrees and many variations have been advanced.

The Agile software development approach called Feature Driven Development (FDD), introduced earlier, describes how a project is divided into "features," which are small pieces of the project that possess some customer value. This lifecycle creates design, code, and code inspection schedules that may seem strangely un-agile, but these schedules lack the depth and mounds of paperwork associated with a system completely specified in the requirements phase, instead relying on people and their roles to address the details as needed. Palmer et al [5] discuss the best practices that make up FDD:

- Domain Object Modeling
- Developing by Feature
- Individual Class (Code) Ownership
- Feature Teams
- Inspections
- Regular Builds
- Configuration Management
- Reporting/Visibility of Results

Palmer et al [5] also sum up FDD in four sentences. "FDD starts with the creation of a domain object model in collaboration with Domain Experts. Using information from the modeling activity and from any other requirements activities that have taken

place, the developers go on to create a features list. Then the rough plan is drawn up and responsibilities are assigned. Now we are ready to take small groups of features through a design and build an iteration that lasts no longer than two weeks for each group and is often much shorter, repeating this process until there are no more features." This approach is depicted below, see Figure 2.

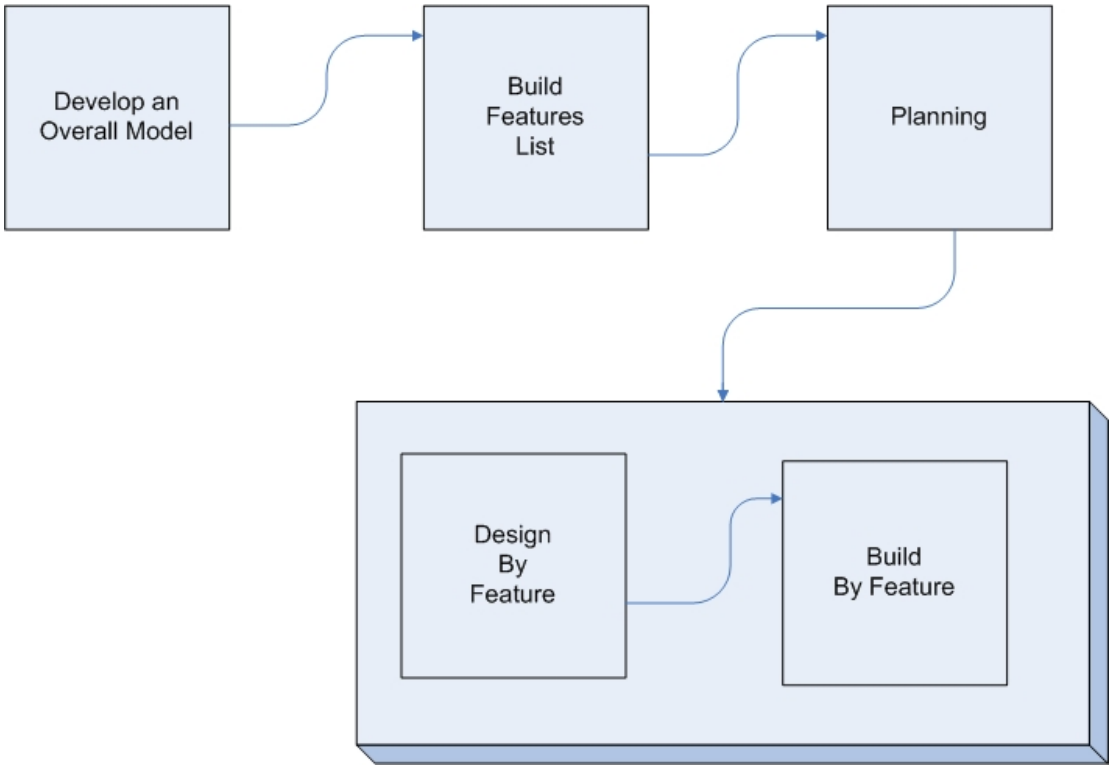


Figure 2 FDD Lifecycle Representation

In this representation, the overall model is thought to provide more shape than content. Also, the feature list is grouped into sets and subject areas.

The concept of quantifying software development and software attributes is necessary for measuring the reliability, quality and safety of a software application. To assist in quantification, an understanding of the best practices employed in

software engineering is essential. The best practices of software engineering, as discussed by Jones [14], are affected by:

- Size of the application
- Type of software (embedded, web, military, systems, etc.)
- Activity (development, deployment and maintenance)

The process of best practices attempts to chart a set of best practices for a software development to follow. Many times these paths are based on size and type of application. These paths contain many of the metrics used for quantifying software reliability and if followed can provide a team with metrics to use in software quality improvement. Jones [14] provides a chart, in the figure below, which can be used for teams to provide best practices for their software development.

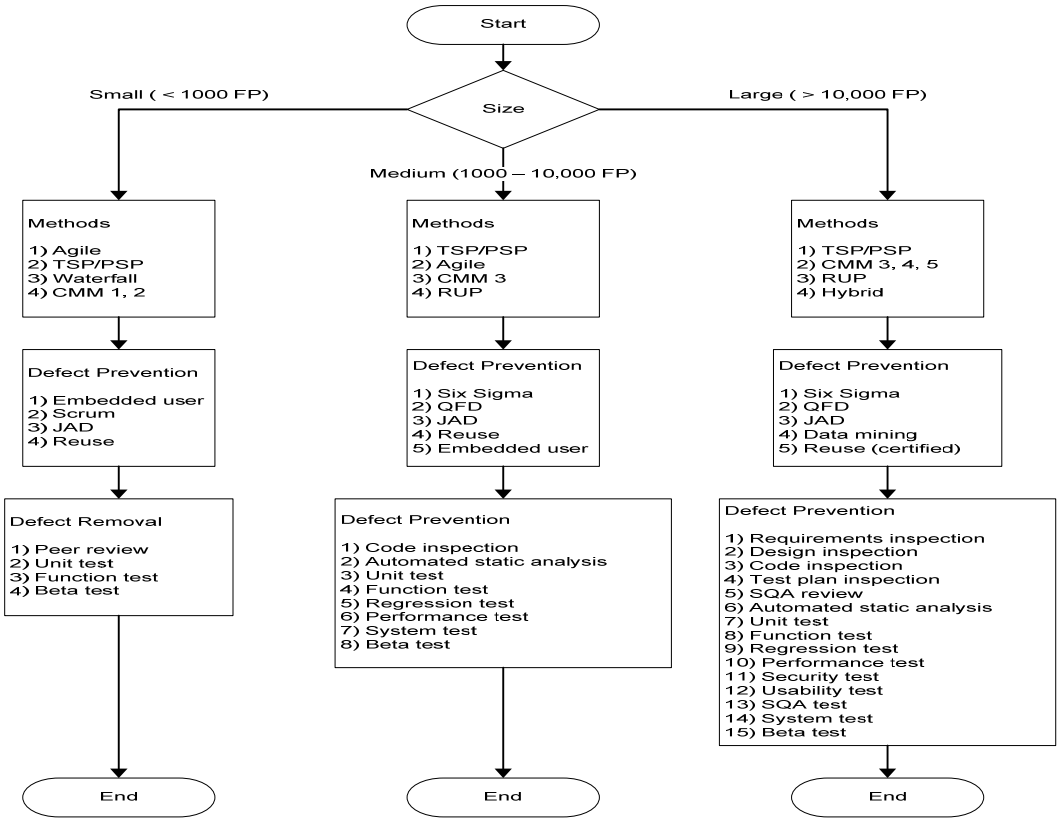


Figure 3 Software Development Best Practices by Application Size

These software best practices, according to Jones [14], provide an outline in which software metrics provide the best and most accurate measurements to use in software modeling.

1.4.4 A Stochastic Model of Fault Introduction & Removal During Software Development

Stutzke and Smidts [12] have introduced a software reliability model incorporating the aspects of stochastic behavior relating to software development and error occurrence in the software development phases. The model is applied to early life-cycle phases. The model is used to capture a stochastic process of software development consisting of fault introduction, from development and debugging errors, and fault removal, from error detection followed by repair. The model assumes that the development errors follow a nonhomogeneous Poisson process and that the fault count at any time in the system is described as a birth-death-with-immigration (BDWI) process and that multiple births and deaths may occur. Thus, the software reliability model is described as being a nonhomogeneous, multiple, birth-death with immigration (NHMBDWI) process.

This model is applied to two different software development lifecycles, namely, Waterfall and Agile Feature Driven Iterative Development. The graphical representation of these lifecycles is presented in this research. The derivation and mapping of these lifecycles is shown using Markov models. The two lifecycles are then contrasted using the Markov models. The models, adapted to their lifecycles, are then implemented with fault data gathered from two industry projects, one per the

previously mentioned lifecycles and industry metrics. The data is analyzed to verify and optimize the model in both lifecycle domains.

Possible optimizations and tuning processes are suggested for the model. The Success Likelihood Index (SLI) is discussed in detail. Other possible parameters, in both the fault introduction and fault removal phases, which affect the accuracy and optimization of the model are discussed and considered in a context of real implementation of the model.

1.5 Research Contributions

The research contributions by the author of the thesis are as follows:

1) The author performed the extension of the Stochastic Model of Fault Introduction & Removal During Software Development [12] to the Feature Driven Development software development process. To accomplish this objective, extensions to the Function Point Analysis (FPA) [9] process as well as extensions to the model were performed. This involved consultations with FPA experts and research into the consequences to FPA when extending the function point counting rules to the FDD process. As a consequence, a small set of new counting rules were developed and validated by an expert in the FPA area. These new counting rules have been characterized as "local counting rules" and pertain to an iterative development lifecycle, such as FDD, and were developed in this research.
2) Extending the Stochastic Model of Fault Introduction & Removal During Software Development [12] to the FDD process involved mapping of new fault introduction and removal phases into the model which were not included

in the original model. This is because the original model was developed based on the Waterfall process. The new phases, in the FDD process, include System Model, System Model Review, Feature List development, Feature List Review and Feature List Planning. To incorporate these new phases into the model the characteristics and metrics associated with the phases needed to be developed.

3) A Markov representation of both the Waterfall and FDD lifecycle was developed as a tool in analyzing the transitions between the lifecycle states. The Markov model also provides an overview of the Kolomogorov graphical representation of each lifecycle state.
4) Research into the individual parameters which make up the equations of the model revealed the terms which could be optimized. These are parameters which are not given by industry standard data or calculated from industry standard data. The parameters identified as being able to be optimized, including SLI, are considered to be expert opinion and highly subjective data. All the parameters of the model were considered, which required a vast investigation into the software metrics involved in both lifecycles. A thorough understanding of the model's parameters and their origins was developed through literature reviews and conversations with experts.
5) Another research contribution is the insertion of the SLI parameter to the Fault Removal phases of software development. To accomplish this, the model equations were manipulated such that SLI was inserted into the fault

removal section. Research into Defect Removal Efficiency (DRE) data revealed necessary input parameters to utilize the SLI in fault removal phases.

6) Once the SLI parameters were applied to the fault removal phases an optimization of SLI was attainable. This research developed a software tool to optimize the SLI of a fault introduction phase and its immediately subsequent fault removal phase based on the actual observed and removed faults of the removal phase.
7) Software tools were developed to provide for automated data entry and analysis into the Stochastic Model of Fault Introduction & Removal During Software Development [12].
8) A software application was developed to provide for automatic entry of data, analysis of the given data and reporting of model results for a Waterfall software development process. The data necessary for the model is read from an ASCII file. The data includes the primitive parameters which make up the fault introduction and fault removal phases. The application then processes the data in the model for each phase and displays the results per phase and per fault type. A printed report is available showing the displayed results. This tool provides for immediate feedback of the parameter optimizations which have been applied.
9) The software application described above was also implemented as a tool for the FDD software development lifecycle. The functions are the same, although this application is more complex as more phases are analyzed.

10) Another software application was developed to perform the optimization of the SLI parameter. The resulting equation when solving for SLI contains numerous solutions. Thus, a programmatic approach to choosing the best solution was chosen and implemented. A spreadsheet to graphically display the in-progress SLI optimization was also developed.

Chapter 2: Development of the Reliability Model for Waterfall Development

2.1 The Software Reliability Model

The Stochastic Model of Fault Introduction & Removal During Software Development [12] as applied to a waterfall lifecycle uses Kolomogorov equation development to develop the equations used for software reliability prediction.

The model, at a high level, is shown in Figure 4, and represents the action of fault injection and removal during generalized software development.

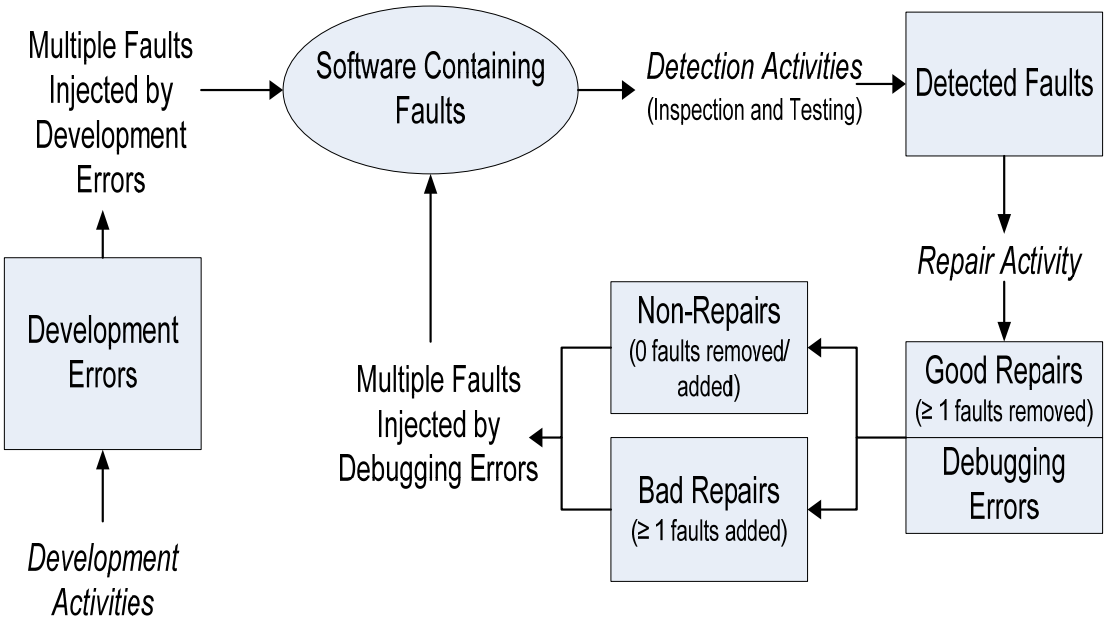


Figure 4 Injection and Removal of Software Faults in a Software Development Lifecycle

2.1.1 Non-Homogenous Multiple Birth-Death-With-Immigration Process

Representing the lifecycle as a Non-Homogenous Multiple Birth-Death-With-Immigration (NHMBDWI) process, as described in section 1.4.4, makes certain assumptions [12]. Among these assumptions are that development errors follow a non-homogeneous Poisson process (NHPP) with an intensity function $\nu(t)$. The value $\nu(t)$ is a function of time, thus, the definition as an NHPP. These development errors add at least one fault to the software. The multiple software fault count is described by a birth-death-with-immigration (BDWI) process, which obeys the Markov property. A stochastic process obeys the Markov property if when the present state is known, the probability of any particular future behavior of the process is not altered by additional knowledge about its past behavior [18]. Faults are detected either by inspections or dynamic testing. Software fault detection follows an NHPP. Software failure is caused by one, and only one, fault at any instant of time. When a fault is detected, a repair process is undertaken resulting in one of three outcomes; a good repair, a non-repair or a bad repair.

The NHMBDWI process is non-homogeneous because the fault intensity function, $\nu(t)$, and the software fault detection rate, $z_a(t)$, are time-dependent. The process relates to faults in multiple ways, as greater than or equal to one fault are added due to development errors or bad repair errors and greater than or equal to one error is also removed due to good repairs. The immigration of faults occurs in development errors; they are born due to bad repairs and experience death due to good repairs.

The NHMBDWI process uses a discrete-state Markov model to represent the software fault count. The stochastic process is described using Kolmogorov forward equations [12].

2.1.2 NHMBDWI Process Model Development for the Waterfall Lifecycle

To model the NHMBDWI process for the Waterfall lifecycle the Kolmogorov forward equations describe the lifecycle for both fault introduction and fault removal phases [12].

2.1.2.1 Model for Development of the Kolmogorov Forward Equations

The Kolmogorov forward equations use the following notation to aid in describing the stochastic process of software development [12].

- $\nu(t) \rightarrow$ rate of occurrence of development errors
- $\mu_H(t) \rightarrow$ s-expected number of faults added by one development error
- $h_{k|i}(t) \rightarrow$ the probability that a development error occurring at time t adds k faults to the software when there were i faults
- $z_a(t) \rightarrow$ intensity function of per fault detection
- $\mu_R(t) \rightarrow$ s-expected change in fault count due to one repair
- $\mu_U(t) \rightarrow$ s-expected number of faults at time t
- $\beta_{k|i}(t) \rightarrow$ the probability that k faults are added given a fault is detected at time t when there were i faults
- $\gamma_{k|i}(t) \rightarrow$ the probability that k faults are removed given a fault is detected at time t when there were i faults
- Pr{good repair at time t} = $g_i(t) = \sum_{k=1}^{i} \gamma_{k|i}(t)$
- Pr{bad repair at time t} = $b_i(t) = \sum_{k=1}^{\infty} \beta_{k|i}(t)$

- Pr{non-repair at time t} = 1 - g_i (t) - b_i (t)

The state transition diagram for the fault count is given in Figure 5.

It describes transitions between states due to the effect of ν (t); the rate of occurrence of errors, z_a (t); the intensity function of per fault detection, $\beta_{k \mid i}(t)$; the probability that k faults are added given a fault is detected at time t when there were i faults, and $\gamma_{k \mid i}(t)$; the probability that k faults are removed given a fault is detected at time t when there were i faults. As state transitions occur from 0 to (i + 1) the dynamic nature of fault introduction, through development errors or bad repairs and fault removal, through good repairs is shown. When development errors are present and a review or inspection detects these errors, a repair is attempted with three possible outcomes:

- Good Repair: ≥ 1 faults are removed ($\gamma_{k \mid i}(t)$)
- Non-Repair: No faults are removed (Self transitions are not shown)
- Bad Repair: ≥ 1 faults are added ($\beta_{k \mid i}(t)$)

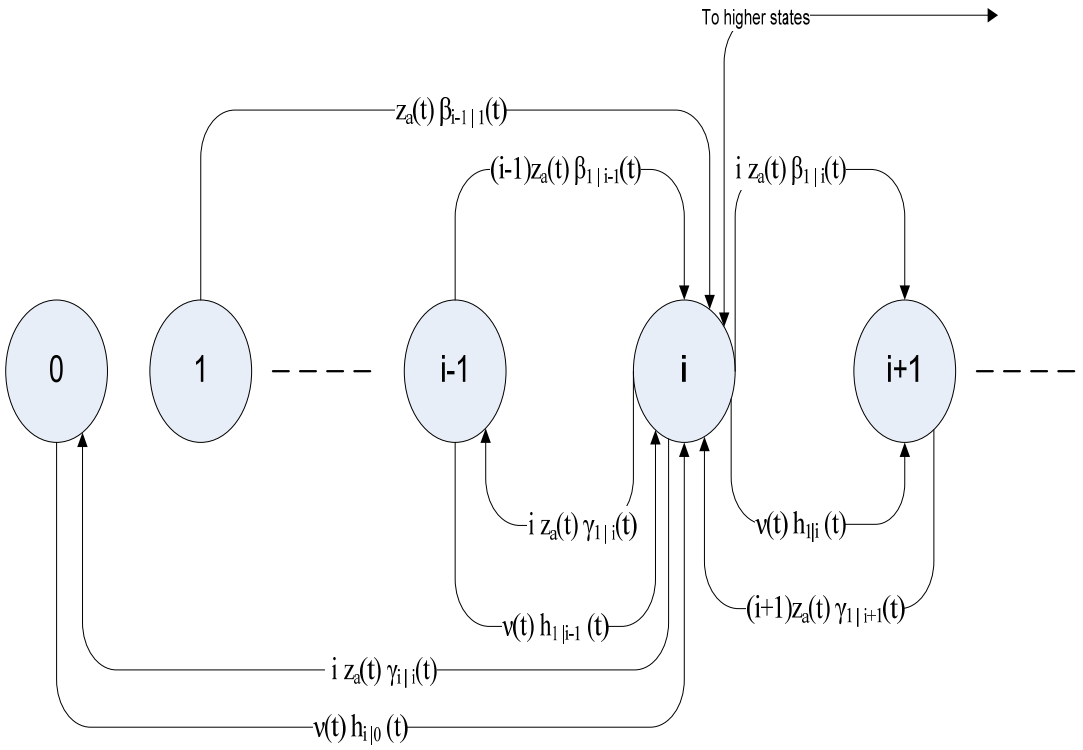


Figure 5 Kolmogorov Forward Equation State Transition Diagram of Development Errors, Bad Repairs, and Good Repairs

In Figure 5, the Kolmogorov Forward Equation state transition diagram shows Development Errors, Bad Repairs, and Good Repairs. This provides the basis to acquire the state equations for the model. The final form of the NHMBDWI state (Kolomogorov forward) equations is [12]:

$$\frac{dP_0(t)_{j,\varphi}}{dt} = z_a(t)_{j,\varphi} \bullet \sum_{k=1}^{\infty} k \bullet \gamma(t;k|k)_{j,\varphi} \bullet P_k(t)_{j,\varphi} - \nu(t)_{j,\varphi} \bullet P_0(t)_{j,\varphi} \tag{2.1}$$

$$\begin{aligned}\frac{dP_1(t)_{j,\varphi}}{dt} = {} & z_a(t)_{j,\varphi} \bullet \sum_{k=1}^{\infty}(k+1)_{j,\varphi} \bullet \gamma(t;k|k+1)_{j,\varphi} \bullet P_{k+1}(t)_{j,\varphi} \\ & + \nu(t)_{j,\varphi} \bullet h(t;1|0)_{j,\varphi} \bullet P_0(t)_{j,\varphi} \\ & - \{z_a(t)_{j,\varphi} \bullet \lfloor g_1(t)_{j,\varphi} + b_1(t)_{j,\varphi} \rfloor + \nu(t)_{j,\varphi} \} \bullet P_1(t)_{j,\varphi}\end{aligned} \tag{2.2}$$

$$\frac{dP_i(t)_{j,\varphi}}{dt} = z_a(t)_{j,\varphi} \bullet \sum_{k=1}^{i-1} k \bullet \beta(t : i-k|k)_{j,\varphi} \bullet P_k(t)_{j,\varphi} \qquad (2.3)$$
$$+ z_a(t)_{j,\varphi} \bullet \sum_{k=1}^{\infty} (k+i)_{j,\varphi} \bullet \gamma(t; k+i)_{j,\varphi} \bullet P_{k+i}(t)_{j,\varphi}$$
$$+ \nu(t)_{j,\varphi} \bullet \sum_{k=0}^{i-1} h(t; i-k|k)_{j,\varphi} \bullet P_k(t)_{j,\varphi}$$
$$- \{i \bullet z_a(t)_{j,\varphi} \bullet \lfloor g_i(t)_{j,\varphi} + b_i(t)_{j,\varphi} \rfloor + \nu(t)_{j,\varphi}\} \bullet P_i(t)_{j,\varphi}, i \geq 2$$

Where:

j	a category of faults introduced during phase j, j = RQ, DE or CO
φ	a lifecycle phase, φ = RQ, RR, DE, DR, CO, CI, UT, IgT, ST, AT
$\nu(t)_{j,\varphi}$	rate of occurrence of development errors of type "j" in phase φ
$h(t;k\|i)_{j,\varphi}$	the probability that a development error occurring at time t adds k faults to the software when there were i faults of type "j" in phase φ
$z_a(t)_{j,\varphi}$	per fault detection intensity function for faults of type "j" in phase φ
$\beta(t;k\|i)_{j,\varphi}$	the probability that k faults are added given a fault is detected at time t when there were i faults of type "j" in phase φ
$\gamma(t;k\|i)_{j,\varphi}$	the probability that k faults are removed given a fault is detected at time t when there were i faults of type "j" in phase φ
$P_i(t)_{j,\varphi}$	Pr{ *i* = fault content at *t* of type "j" in phase φ }
$b_i(t)_{j,\varphi}$	Pr{ a bad repair at *t* | *i* faults in the software of type "j" in phase φ }

$g_i(t)_{j,\varphi}$ Pr{ a good repair at t | i faults in the software of type "j" in phase φ }

An equation for $\mu_u(t)_{j,\varphi}$, the s-expected number of faults of type "j" at time t in phase "φ", can be obtained by making two assumptions [12]. First, the number of faults injected into the software due to development errors does not depend on the number of faults present at the time when the development error occurs; thus:

$$h(t;k|i)_{j,\varphi} \rightarrow h(t;k)_{j,\varphi} \tag{2.4}$$

Second, the s-expected change does not depend on i, at the time when a repair is attempted; thus:

$$\mu_R(t;i)_{j,\varphi} \rightarrow \mu_R(t)_{j,\varphi} \tag{2.5}$$

Where:

$\mu_R(t;i)_{j,\varphi}$ is the s-expected change in fault count of type "j" in phase φ due to one repair

The solution of the NHMBDWI state equations (2.1) to (2.3) is not trivial [12]. The Kolmogorov Forward Equation models the expected number of faults in the software as a function of time [12]. Using assumptions (2.4) and (2.5) we obtain equation (2.6).

$$\frac{d\mu_U(t)_{j,\varphi}}{dt} = \{\nu(t)\mu_H(t)\}_{j,\varphi} + \{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}\{\mu_U(t)\}_{j,\varphi} \tag{2.6}$$

Where:

$\{\mu_U(t)\}_{j,\varphi}$ the s-expected number of faults of type "j" at time t in phase φ

j	a category of faults introduced during phase j, ie., j = RQ, DE or CO
φ	a lifecycle phase, ie., φ = RQ, RR, DE, DR, CO, CI, UT, IgT, ST, AT
t	software development lifecycle time
$\{\nu(t)\ \mu_H(t)\}_{j,\varphi}$	Estimate of "j" fault introduction rate in phase φ
$\{z_a(t)\}_{j,\varphi}$	intensity function of per-fault detection in phase φ, for type "j" faults
$\{\mu_R(t)\}_{j,\varphi}$	the s-expected change in fault count of type "j" in phase φ due to one repair

The model consists of three parameters. The first parameter is the unadjusted estimate of the fault-introduction rate of the j-th fault categories, $\{\nu(t)\ \mu_H(t)\}_{j,\varphi}$. The second parameter is the expected change in fault count due to one repair, $\{\mu_R(t)\}_{j,\varphi}$. The third parameter is the intensity function of per-fault detection in phase φ, for category "j" faults, $\{z_a(t)\}_{j,\varphi}$. The term $\{\nu(t)\ \mu_H(t)\}_{j,\varphi}$ addresses the introduction of faults in a software development process while the term, $\{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}\{\mu_U(t)\}_{j,\varphi}$ addresses the detection and removal of faults.

2.1.3 Estimate of the Fault Introduction Rate

In this lifecycle, faults are introduced in the RQ, DE and CO phases. The fault introduction rate within a phase is constant and is given by:

$$\{\nu(t)\bullet\mu_H(t)\}_{j,\varphi}=\begin{Bmatrix}\{\overline{\nu(t)\bullet\mu_H(t)}\}_{j,\varphi}\bullet F_{j,\varphi}^{1-2SLI_\varphi} & ;\varphi=j\\ 0 & ;\varphi\neq j\end{Bmatrix} \tag{2.7}$$

and

$$\left\{\overline{\nu(t) \bullet \mu_H(t)}\right\}_{j,\varphi} = \frac{DP_{\varphi} \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \qquad (2.8)$$

Where:

$\left\{\overline{\nu(t) \bullet \mu_H(t)}\right\}_{j,\varphi}$	unadjusted estimate of the fault introduction rate of the j-th fault categories in phase φ
j	a category of faults introduced during phase j, j = RQ, DE or CO
φ	a lifecycle phase, ie., φ = RQ, RR, DE, DR, CO, CI, UT, IgT, ST, AT
$F_{j,\varphi}$	a constant for type "j' faults in phase φ
SLI_{φ}	Success Likelihood Index which varies between 0 (error is likely) and 1 (error is not likely) in phase φ
DP_{φ}	defect potential per function point in phase φ
$fd_{j,\varphi}$	Fraction of faults of type "j" that originated in phase φ
$\bar{t}_{FP,\varphi}$	mean effort necessary to develop a function point in phase φ

In Equation 2.7, SLI_{φ} is a parameter which is not taken from industry standards or calculated from the industry standards; instead it is a parameter which is assessed for the specific software development under study. It characterizes the specific state of the factors in play during the development process and the degree to which they depart from average industry conditions. The parameters DP_{φ}, $fd_{j,\varphi}$ and $t_{FP,\varphi}$ are based on industry values obtained from the research. The parameter $F_{j,\varphi}$ is the factor that

describes the impact of SLI_φ from 0.5 on ν(t). In Equation 2.7, $\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi} = 0$ while $j \neq \varphi$. This is because it is assumed that each category of faults is only introduced in its corresponding development phase. For example, the Requirements faults (j = RQ) are introduced in the Requirements phase (φ = RQ). Therefore, the introduction of the Requirements Faults is 0 during other development phases (φ = DE or CO) or other removal phases. For $F_{j,\varphi}$, the following transformations should be made. The upper and the lower bounds on $\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi}$ (corresponding to the extreme values of SLI: 0, worst case development conditions and 1, best case development conditions) are:

$$\{(\nu(t) \bullet \mu_H(t))_{\min}\}_{j,\varphi} = \frac{1}{F_{j,\varphi}}\left(\overline{\nu(t) \bullet \mu_H(t)}\right)_{j,\varphi} \quad \rightarrow SLI_\varphi = 1, \tag{2.9}$$

$$\{(\nu(t) \bullet \mu_H(t))_{\max}\}_{j,\varphi} = F_{j,\varphi}\left(\overline{\nu(t) \bullet \mu_H(t)}\right)_{j,\varphi} \quad \rightarrow SLI_\varphi = 0, \tag{2.10}$$

Stutzke [12] proposes a method for estimating $F_{j,\varphi}$, in [23] this method is illustrated. This proposed method uses equations 2.9 and 2.10 to obtain equation 2.14. To obtain the upper and lower bounds on $\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi}$ in the development phase, the upper and lower bounds of $DP_\varphi \cdot fd_{j,\varphi}$ and $\bar{t}_{FP,\varphi}$ should be obtained first.

The upper bound, lower bound and mean of $\{\nu(t) \bullet \mu_H(t)\}_{j,\phi}$, from equation 2.8 is given by:

$$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi UpperBound} = \frac{DP_{\varphi,MAX} \bullet fd_{j,\varphi\,MAX}}{\bar{t}_{FP,\varphi MIN}} \tag{2.11}$$

$$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi LowerBound} = \frac{DP_{\varphi,MIN} \bullet fd_{j,\varphi\,MIN}}{\bar{t}_{FP,\varphi MAX}} \tag{2.12}$$

$$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi MEAN} = \frac{DP_{\varphi,MEAN} \bullet fd_{j,\varphi\,MEAN}}{\bar{t}_{FP,\varphi MEAN}} \tag{2.13}$$

Therefore, it follows that the value of the constant $F_{j,\varphi}$ is given by:

$$F_{j,\varphi} = \sqrt{\frac{\{(\nu(t) \bullet \mu_H(t))_{\max}\}_{j,\varphi}}{\{(\nu(t) \bullet \mu_H(t))_{\min}\}_{j,\varphi}}} \tag{2.14}$$

2.1.4 Estimate of the Expected Change in Fault Count Due to One Repair

When data for estimating $\{\mu_R(t)\}_{j,\varphi}$ is not available (especially for φ = RQ, DE and CO, in which the debugging activities are rarely documented), -0.7, the industry average [12] [23] should be used.

2.1.5 Estimate of the Fault Removal Rate for Phase of Origin

The fault removal referred to in this section concerns the removal of a "j" type of fault removed in the phase in which it was created. This removal occurs through the process of desk checking the development as it is being created. Examples of desk checking for faults in a phase of origin include [12] [23] detecting:

- requirements faults during the requirements analysis phase
- design faults during a design phase
- coding faults during code development

At time, t, the developer is able to review $(t_\varphi - t_{0,\varphi}) / t_{FP,\varphi}$ function points, where $t_{0,\varphi}$ is the time at which the considered phase, φ, originates, and this amount of development now exists. The developer reviews the entire development built during $(t_\varphi - t_{0,\varphi})$ as $\{(t_\varphi - t_{0,\varphi}) / t_{FP,\varphi}\}$ origin function points. During that time, the developer can observe all k_φ (number of origin) faults present in the development of that phase. Only $x_\varphi \cdot k_\varphi$ of the k_φ faults are actually detected, where x_φ is the fault detection efficiency, hence the equation 2.15.

$$k_\varphi \bullet z_a(t)_{j,\varphi} = \frac{k_\varphi \bullet r_\varphi \bullet x_\varphi}{\frac{(t_\varphi - t_{0,\varphi})}{t_{FP,\varphi}}} \qquad (2.15)$$

Where:

$(t_\varphi - t_{0,\varphi}) / t_{FP,\varphi}$	number of origin function points developed by time $(t - t_0)$ in phase φ
$z_a(t)_{j,\varphi}$	per fault detection intensity function for faults of type "j" in phase φ
r_φ	number of function points that a developer can check per unit of time in phase φ
x_φ	fault-detection efficiency in phase φ
$t_{FP,\varphi}$	effort necessary to develop a function point in phase φ
t_φ	time
$t_{0,\varphi}$	t at which the considered phase originates
k_φ	number of origin faults in the software at time t in phase φ

Next, we obtain an approximate of the number of function points developed by the factor: $\{(t_\varphi - t_{0\varphi}) / t_{FP,\varphi}\}$, which is an upper bound on the real number of function points

created. This is an upper bound due to the fact that every time the developer interrupts development to check work, the creative development process slows down; N_{FP} (the number of function points) is overestimated and the per-fault failure intensity is under-estimated.

$$z_a(t)_{j,\varphi} = \frac{r_\varphi \bullet x_\varphi \bullet t_{FP,\varphi}}{t_\varphi - t_{0,\varphi}} \tag{2.16}$$

The "maximum number of origin-faults" present is:

$$(\text{Number of faults created})_{j,\varphi} = \nu_{j,\varphi} \cdot \mu_{H,j,\varphi} \cdot T_\varphi \tag{2.17}$$

Where:

$\nu_{j,\varphi}$ rate of occurrence of development "j" type faults in phase φ

$\mu_{H,j,\varphi}$ s-expected number of "j" faults added by 1 development error in phase φ

T_φ length of a phase φ

$\nu_{j,\varphi} \cdot \mu_{H,j,\varphi}$ is constant over the phase. Then $\rho(t)_{j,\varphi}$, the number of 'j' type faults removed during a development phase φ, is determined from:

$$\frac{d\rho_{j,\varphi}}{dt} = z_a(t)_{j,\varphi} \bullet \mu_U(t)_{j,\varphi} \tag{2.18}$$

Where:

$\mu_U(t)_{j,\varphi}$ s-expected number of"j" type faults at t in phase φ

and is in parallel with $\mu_{U,j,\varphi}$. We are interested in the fault-detection capability of the development process and not in the efficiency or characteristics of the repair process: thus $\{\mu_R(t)_{j,\varphi} \equiv 1$, $\{\mu_R(t)\}_{j,\varphi}$ is the s-expected change in fault count due to 1 repair in equations governing ρ and μ_U. Hence:

$$\rho(t)_{j,\varphi} = \frac{\alpha_\varphi \bullet \nu_{j,\varphi} \bullet \mu_{H,j,\varphi}(t_\varphi - t_{0,\varphi})}{\alpha_\varphi + 1} \tag{2.19}$$

Where:

$\alpha_\varphi = r_\varphi \cdot x_\varphi \cdot t_{FP,\varphi}$

Equation 2.19 leads at the end of the phase to:

$$\frac{NumberOfFaultsremoved_\varphi}{NumberOfFaultsCreated_\varphi} = \frac{\alpha_\varphi}{\alpha_\varphi + 1} \tag{2.20}$$

Where:

$$\frac{NumberOfFaultsremoved_\varphi}{NumberOfFaultsCreated_\varphi} = Defectremovalefficiency_\varphi = DRE_\varphi \tag{2.21}$$

Thus, for fault removal in the phase of origin:

$$z_a(t)_{j,\varphi} = \frac{r_\varphi \bullet x_\varphi \bullet t_{FP,\varphi}}{t_\varphi - t_{0,\varphi}} = \frac{\alpha_\varphi}{t_\varphi - t_{0,\varphi}} = \frac{\frac{DRE_\varphi}{1 - DRE_\varphi}}{t_\varphi - t_{0,\varphi}} \tag{2.22}$$

2.1.6 Estimate of the Fault Removal Rate for Removal Phases

The fault removal referred to in this section concerns the removal of a "j" type of fault that originated in an earlier phase, within a removal phase. This removal occurs through the process of defect removal activities. The fault-detection rate during a removal phase is a function of: N_{FP} (number of function points in a phase), r (fault-rate detection), k (number of origin faults) and x (fault-detection efficiency).

The primary difference with the z_a parameter of the origin phases is that the number of function points is constant and equal to N_{FP}. Recall equation 2.6:

$$\frac{d\{\mu_U(t)\}_{j,\phi}}{dt} = \{\nu(t) \bullet \mu_H(t)\}_{j,\phi} + \{z_a(t)\}_{j,\phi} \bullet \{\mu_R(t)\}_{j,\phi} \bullet \{\mu_U(t)\}_{j,\phi}$$

In a fault removal phase, equation 2.6 becomes:

$$\frac{d\{\mu_U(t)\}_{j,\varphi}}{dt} = \{z_a(t)\}_{j,\varphi} \bullet \{\mu_U(t)\}_{j,\varphi} \tag{2.23}$$

Because;

$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi} = 0$, during a phase other than the phase of origin there is not fault introduction through new development errors and $\mu_R = 1$.. Thus, solving Equation 2.23 for the removal phase yields:

$$\mu_U(t)_{j,\varphi} = \mu_U\left(t_\varphi\right)_{j,\varphi} \bullet \exp\left[-\int_{t_\varphi}^{t} z_a(t)_{j,\varphi}\, dt\right] \tag{2.24}$$

$$= \mu_U\left(t_\varphi\right)_{j,\varphi} \bullet \exp\left[\int_{t_\varphi}^{t} \ln\left[1 - DRE_\varphi\right] \bullet \frac{1}{t_{FP,\varphi} \bullet N_{FP}} dt\right]$$

$$= \mu_U\left(t_\varphi\right)_{j,\varphi} \bullet \left[1 - DRE_\varphi\right]$$

Where:

t is an arbitrary point of the removal phase φ

t_φ is for the beginning of the removal phase φ

$t_{FP,\varphi}$ effort necessary to develop a function point in phase φ

DRE_φ Defect Removal Efficiency in phase φ

N_{FP} Number of function points

$\mu_U(t)_{j,\varphi}$ s-expected number of 'j' type faults at t in phase φ

Integrating equation 2.24, we obtain the proportion of faults which are removed from the software after inspection if repair is perfect as:

$$\frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}} = 1 - \exp\left[-\frac{r_\varphi \bullet x_\varphi \bullet T_\varphi}{N_{FP}}\right] \tag{2.25}$$

Where:

$z_a(t)_{j,\varphi} = r_\varphi \cdot x_\varphi / N_{FP}$

r_φ number of function points that a reviewer / tester can check / test per unit of time in phase φ

x_φ fault-detection efficiency in phase φ

T_φ length of a phase φ

N_{FP} number of function points

$\mu_U(t)_{j,\varphi}$ s-expected number of faults at t

Then,

$$x_\varphi = -\ln\left[1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \bullet \frac{N_{FP}}{r_\varphi \bullet T_\varphi} \tag{2.26}$$

$$z_a(t)_{j,\varphi} = -\ln\left[1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \bullet \frac{1}{T_\varphi} \tag{2.27}$$

Because,

$$\exp\left[-\frac{r_\varphi \bullet x_\varphi \bullet T_\varphi}{N_{FP}}\right] = 1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T_\varphi)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}} \tag{2.28}$$

$$-\frac{r_\varphi \bullet x_\varphi \bullet T_\varphi}{N_{FP}} = \ln\left[1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T_\varphi)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \tag{2.29}$$

$$z_a(t)_{j,\varphi} \bullet T_\varphi = -\ln\left[1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T_\varphi)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \tag{2.30}$$

$$z_a(t)_{j,\varphi} = -\ln\left[1 - \frac{\mu_U(t_\varphi)_{j,\varphi} - \mu_U(t_\varphi + T_\varphi)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \bullet \frac{1}{T_\varphi} \tag{2.31}$$

$$z_a(t)_{j,\varphi} = -\ln\left[\frac{\mu_U(t_\varphi + T_\varphi)_{j,\varphi}}{\mu_U(t_\varphi)_{j,\varphi}}\right] \bullet \frac{1}{T_\varphi} = -\ln\left[1 - DRE_\varphi\right] \bullet \frac{1}{t_{FP,\varphi} \bullet N_{FP}} \tag{2.32}$$

2.1.7 The Success Liklihood Index

Software development is affected by human reliability [12] since all development activities are carried out by humans. Human reliability is affected by factors relating to a developer's environment or task and can affect performances positively or negatively [19]. These factors are called influencing factors and affect the magnitude of the intensity and probability density functions of fault introduction, detection and removal. A method to account for the quantitative aspects of influencing factors is the success likelihood index method (SLIM). SLIM is founded on three key assumptions [29] [30] [31]:

1. The likelihood of an error occurring in a particular situation depends on the combined effects of a relatively small number of performance influencing factors (PIFs), which are represented by an index (the success likelihood index (SLI)) that ranges from 0 (error is likely) to 1 (error is not likely).
2. An expert panel is asked to identify the value between 0 and 1 of these SLIs in certain situations.
3. The probability of a human error is logarithmically proportional to the SLI [23].

   SLI = a ln HEP + b

   Where HEP = Human Error Probability

The influencing factors used in this research are experience, capability, software complexity, schedule pressure, use of methods / notations, communications, team relationships, management style, process integration method and requirements volatility. The influencing factors are ranked as multiples of the least important, r_i, which is set to 10. The normalized ranks are computed, n_i. Each influencing factor is assessed for quality from best (1) to worst (0), q_i. The SLI can then be calculated as the dot product of the normalized ranks and the qualities.

Table 2 Influencing Factors

Rank	Normalized Rank	Influencing Factor
110	0.204	experience (novice / expert)
20	0.037	capability (low / high)
80	0.148	software complexity (difficult / simple)
50	0.093	schedule pressure (heavy / light)
30	0.056	use of methods/notations (crude / refined)
90	0.167	communications (poor / good)
10	0.019	team relationships (poor / good)
20	0.037	management style (intrusive / supportive)
40	0.074	process integration method (crude / refined)
90	0.167	requirements volatility (chaotic / stable)
540	1	TOTAL

2.1.8 Criticality of Faults

Faults are observed in review, inspection or testing phases. The severity of these faults may be categorized as 1 through 5. A category 1 or 2 fault is considered the most critical or severe and would result in incorrect operation of the device [14]. A category 3 or 4 fault is a minor or cosmetic problem [14]. A category 5 fault is an enhancement not yet implemented, non-conformance to a standard or non- pleasing

aesthetic of the system [14]. The faults recorded in this research are only of a category 1 or category 2 severity. Criticality can be rated in more than one way and for more than one purpose [20] [23], below is an example.

Table 3 Criticality Categories

Category of fault severity	Description		Examples
1	Critical – Critical problem (software does not operate at all)		Missing \ Wrong Requirement, Design element missing, Coding fault
2	Significant – Significant problem (major feature disabled or incorrect)		Enhancement, No Longer Applicable, Removed from the system, etc.
3	Minor – Minor problem (some inconvenience for the users)		
4	Cosmetic – Cosmetic problem (spelling error in messages: no effect on operation)		Typos, Grammar, etc.
5	Exception - The fault is the result of non-conformance to a standard, is related to the aesthetics of the system, or is a request for an enhancement. Faults at this level may be deferred or even ignored.		
	Error Types	• An error of omission (EOO), the failure to carry out some of the actions necessary to achieve a desired goal. • An error of commission (EOC), the carrying out of an unrelated action which prevents the achievement of the goal.	

2.2 Waterfall Lifecycle Description

Described in this section is the use of a waterfall software development lifecycle to verify the stochastic model. The waterfall lifecycle used in this research consists of a Requirements (RQ), Requirements Review (RR), Design (DE), Design Review (DR), Coding (CO), Code Inspection (CI), Unit Test (UT), Integration Test (IgT), System Test (ST) and Acceptance Test (AT) phase. These phases are

characteristic of the traditional waterfall lifecycle depicted in the literature and introduced previously in section 1.4.3.

The requirements phase (RQ) documents the high level user needs and system requirements that the system needs to perform to satisfy the operation of the system. The requirements review (RR) phase gathers a group of project team and independent members to review the given requirements for faults. Once observed, these faults are then corrected and re-reviewed until correctness is ensured. The design phase (DE) consists of the activities necessary to document how the system will convert what is required of the system into an input to the implementation phase of the lifecycle. Again, a design review will gather a review group to verify that the design is correct.

The design is next implemented, in the coding phase (CO). In this phase software is created using the requirements and design, which have been created, verified and corrected if necessary. A code inspection phase (CI) ensures correctness of the coding. An inspection group is assembled, in a manner similar to requirements and design reviews, with an emphasis on coding experts. As the coding phase progresses the individual coder will perform tests on units of code as they are developed, this phase is called unit test (UT). When multiple units are fault free, as determined through unit testing, they are combined and integration testing (IgT) is performed to verify that they are operating correctly together; this is repeated to remove faults. When the code has all been integrated the system is tested (ST). In this phase the complete system will be verified as operating correctly and validated to perform all the required tasks to meet user needs. This phase also is repeated to

remove faults. The system is then given to the user or a representative for their acceptance (AT). Again the testing continues until all faults are removed.

Documentation is created in the RQ, DE, and CO phases. Faults are assumed to be introduced only in the RQ, DE and CO phases. The NHMBDWI process applied to this lifecycle details the introduction and immigration of faults through the phases. Faults are assumed to be removed in the RR, DR, CI, UT, IgT, ST and AT phases. In this research, for the waterfall lifecycle project, documentation is kept for faults observed for the RR (requirements faults), DR (design faults) and ST (requirements, design and coding faults) phases.

The waterfall approach is typically used in software projects that are stable, especially those projects with well understood requirements, and where it is possible and likely that designers will be able to fully predict problem areas of the system and produce a correct design before implementation is started. A term used often with a waterfall lifecycle is the "big design up front" approach.

The advantages to this lifecycle include the idea that much time is spent at the start of the project to ensure that requirements and design are correct before continuing onto implementation. This will find faults sooner in the development cycle, when they are less costly to fix. Another advantage of this lifecycle is the documentation created in the phases; these provide material for team members as the project ages and for team members performing maintenance.

The lifecycle also has disadvantages. Despite the claim of early fault detection, at times a fault may proceed through to a later phase, thus causing an expensive fix [5]. Also, the documentation is time-consuming and for projects

requiring less documentation may be burdensome. Waterfall also is inappropriate for projects which have significant user interfaces or requirements which are not well understood. For these projects the Agile approach to development has been introduced.

2.2.1 *Waterfall Lifecycle Phases*

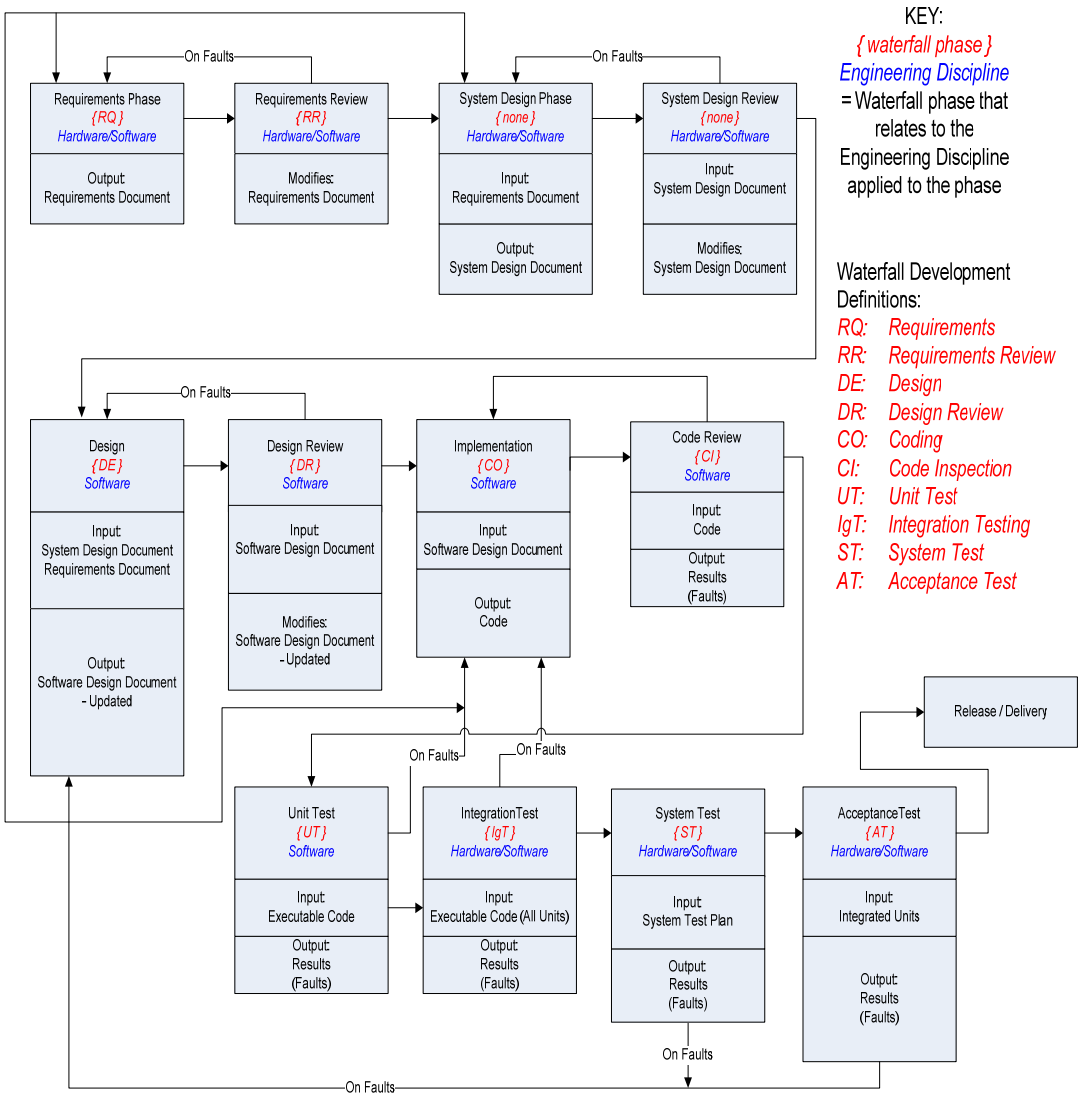


Figure 6 Waterfall Software Development Lifecycle Phases

Each lifecycle phase in Waterfall software development, as implemented in this research and illustrated in Figure 6, has an input, output, next phase and possible fault feedback path for fault removal phases. The phases may create documentation, review documentation, introduce faults, or remove faults. Faults are introduced during phases which design or construct software; these defects are known as

development faults. These faults are categorized as requirements, design or coding defects. Faults may also be introduced during attempts to remove a known defect; these are known as debugging faults. Faults are removed by the processes of reviews, inspections and tests. These phases are known as fault removal phases. This waterfall implementation does not contain all the possible phases contained in the literature concerning waterfall software development.

2.2.1.1 Markov Model of the Waterfall Lifecycle

Transitions between phases of the Waterfall development lifecycle can be modeled as a higher-level Markov model. The model consists of states which represent the fault introduction phases (RQ, DE, CO) and fault removal phases (RR, DR, CI, UT, IgT, ST, AT) and form a Markov chain. Transitions between states are based upon the Dirac Delta (δ) function. The δ function is appropriate because the time spent in each state has been shown to be related to the number of function points (N_{FP}) multiplied by the time per function point (t_{FP}). The δ function is described by:

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

Thus, a transition occurs when the time, t, spent in a state, N_{FP} and $t_{FP,\varphi}$ are related as:

$$\delta(t - (N_{FP} * t_{FP,\varphi})) = 0 \qquad (2.33)$$

The waterfall-like appearance of the lifecycle is maintained in the Markov chain. The Markov chain is depicted in Figure 7. The variables in the diagram are:

t	time spent in a phase
$t_{FP,\varphi}$	time per function point for phase φ
N_{FP}	number of project function points

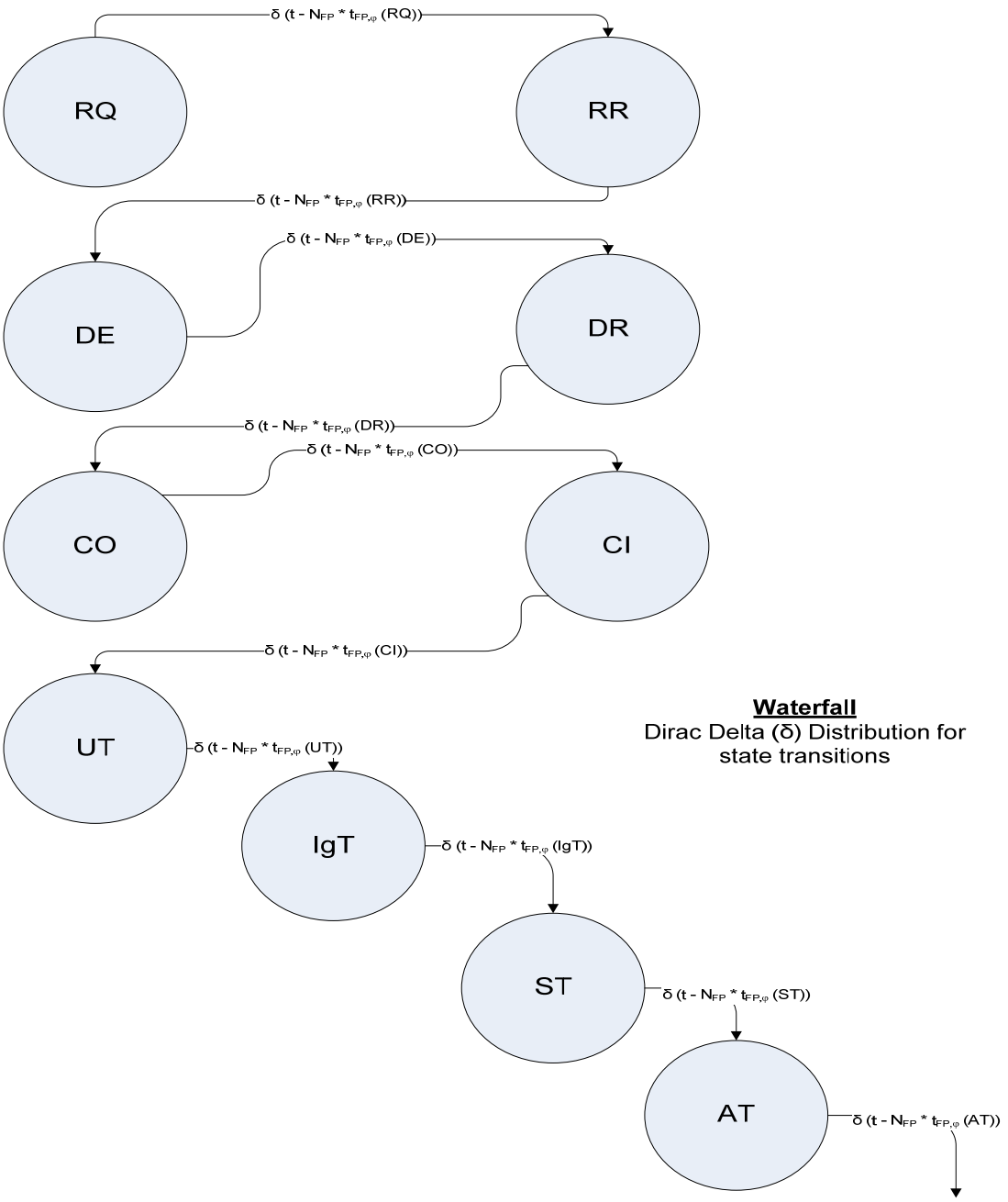


Figure 7 Waterfall Lifecycle Markov Model

Each phase of the Markov model in Figure 7 is decomposed into the lower-level Kolmogorov model representing the fault introduction and removal occurring in that state. The following figures show the expansion of the interior of the Markov chain state as a transition diagram illustrating the fault introduction and removal that

occurs. The effect of fault introduction and fault removal is shown in Figure 5 as a composite model.

In the following figures, depicting the individual waterfall lifecycle phases, the type of fault introduced or removed is defined [12]. Faults are defined by their phase of origin. A requirement (RQ) fault originates in the requirements phase and is reviewed or inspected in subsequent phases. Design (DE) faults originate in the design phase and coding (CO) faults originate in the coding phase and have similar traits with regard to reviews and inspections as RQ faults.

The effect of development errors and debugging errors (bad repairs and non-repairs) along with the good repairs achieved are accounted for inside the phases in the Markov model. In the following figures the type of faults are described as:

- RQ Faults = RQ
- DE Faults = DE
- CO Faults = CO

Where:

$\nu(t)_{\varphi}^{\alpha}$ is the rate of occurrence of development errors of type α in phase φ

$\mu_H(t)_{\varphi}^{\alpha}$ is the s-expected number of faults of type α in added by one development error of type α in phase φ

$h_{k \mid i}(t)_{\varphi}^{\alpha}$ is the probability that a development error occurring at time t adds k faults to the software when there were i faults of type α in phase φ

$z_a(t)^{\alpha}_{\varphi}$ is intensity function of per fault detection of type α in phase φ

$\mu_R(t)^{\alpha}_{\varphi}$ is the s-expected change in fault count of type α due to one repair in phase φ

$\mu_U(t)^{\alpha}_{\varphi}$ is the s-expected number of faults of type α at time t in phase φ

$\beta_{k|i}(t)^{\alpha}_{\varphi}$ is the probability that k faults of type α are added given a fault is detected at time t when there were i faults in phase φ

$\gamma_{k|i}(t)^{\alpha}_{\varphi}$ is the probability that k faults of type α are removed given a fault is detected at time t when there were i faults in phase φ

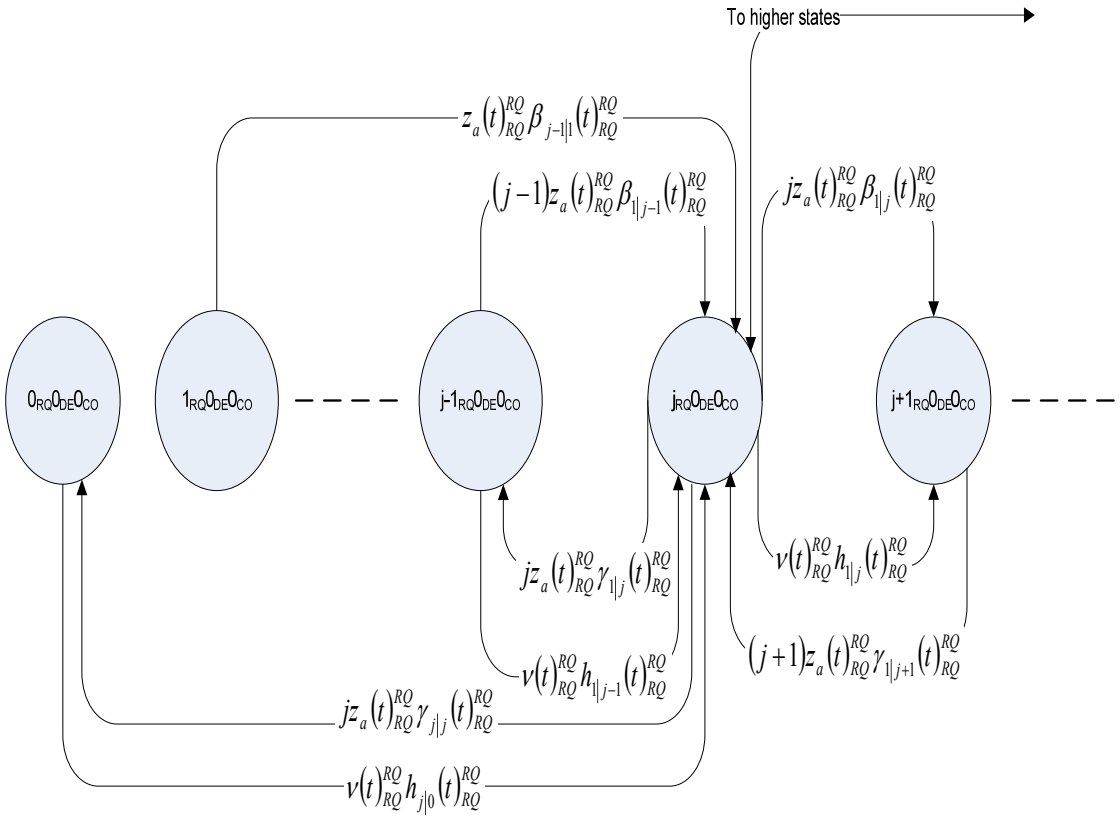


Figure 8 Inside the RQ Phase

The Kolmogorov model represented inside the Markov chain state for the RQ phase (Figure 8) consists of fault introduction and fault removal components. Of the three types of faults, only RQ faults (represented by RQ) will be modified. The other two fault types are not represented as a result of DE and CO faults not being created in the lifecycle as yet.

Fault introduction follows the ν(t) combined with h(t) functions. One fault removal process can actually introduce faults through the introduction of "bad repairs", the $z_a(t)$ combined with β(t) functions. Fault removal is governed by $z_a(t)$ combined with γ(t) functions, "good repairs". The fault introduction and fault removal functions described above are consistent throughout each phase expansion that follows.

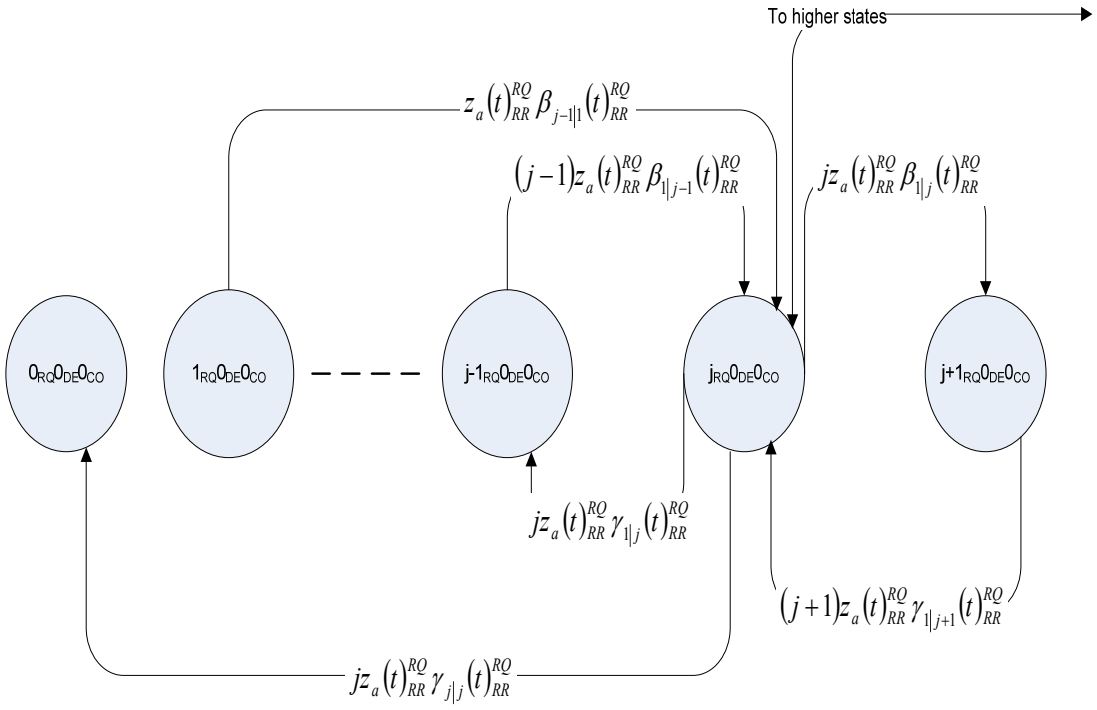


Figure 9 Inside the RR Phase

The Kolmogorov model represented inside the Markov chain state for the RR phase (Figure 9) consists of fault removal components. Of the three types of faults, only RQ faults (represented by RQ) are modified by the corresponding transitions. The other two fault types do not vary as a result of DE and CO faults not being created in the lifecycle as yet. This phase is the primary phase in which requirements faults are removed through requirements reviews.

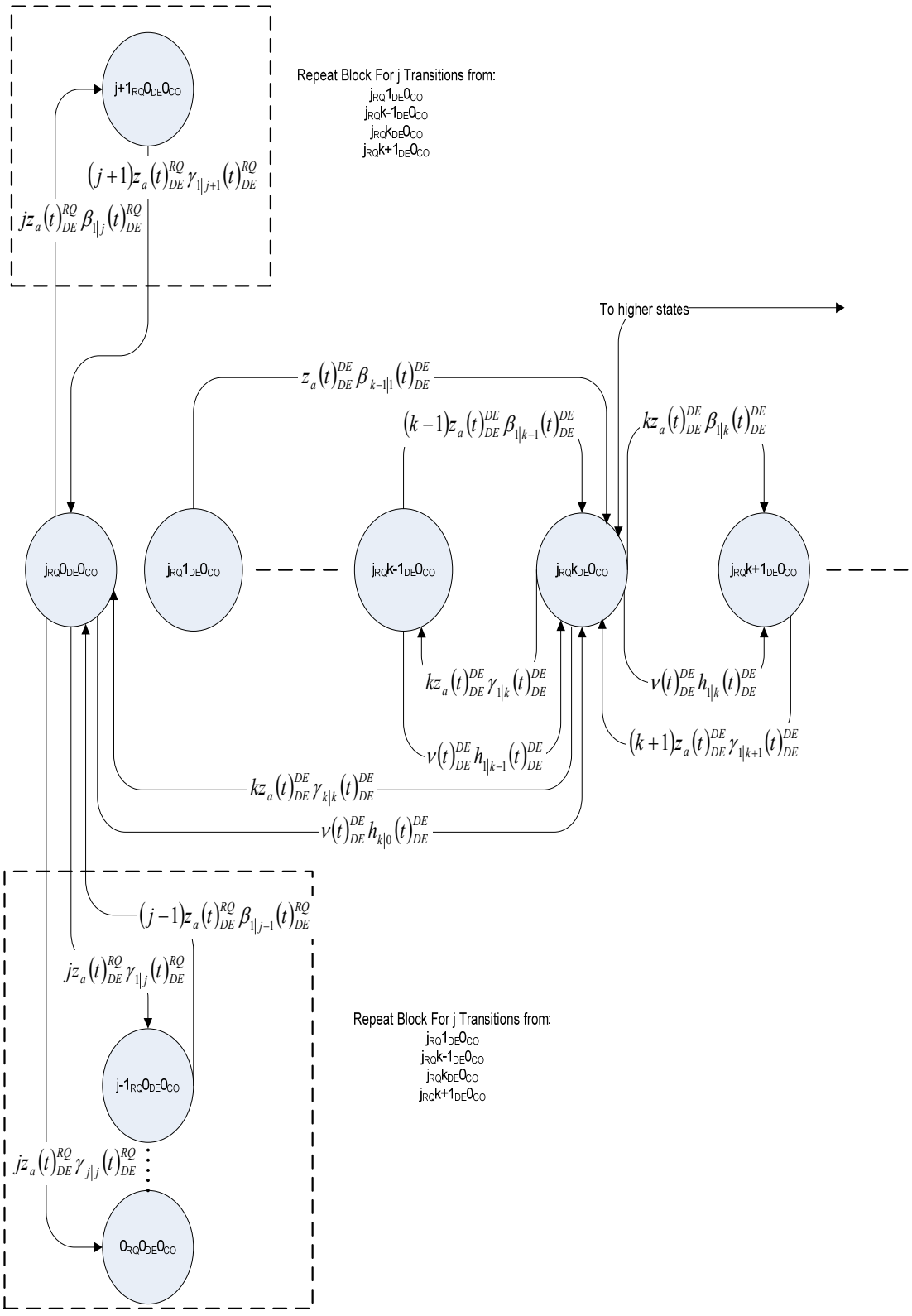


Figure 10 Inside the DE Phase

The Kolmogorov model represented inside the Markov chain state for the DE phase (Figure 10) consists of fault introduction and fault removal components. Of the three types of faults, RQ faults (represented by RQ) and DE faults (represented by DE) will be modified due to applicable transitions. The other fault type will not vary as a result of CO faults not being created in the lifecycle as yet. Since this phase has two active fault types the transitions may occur for either or both types of faults.

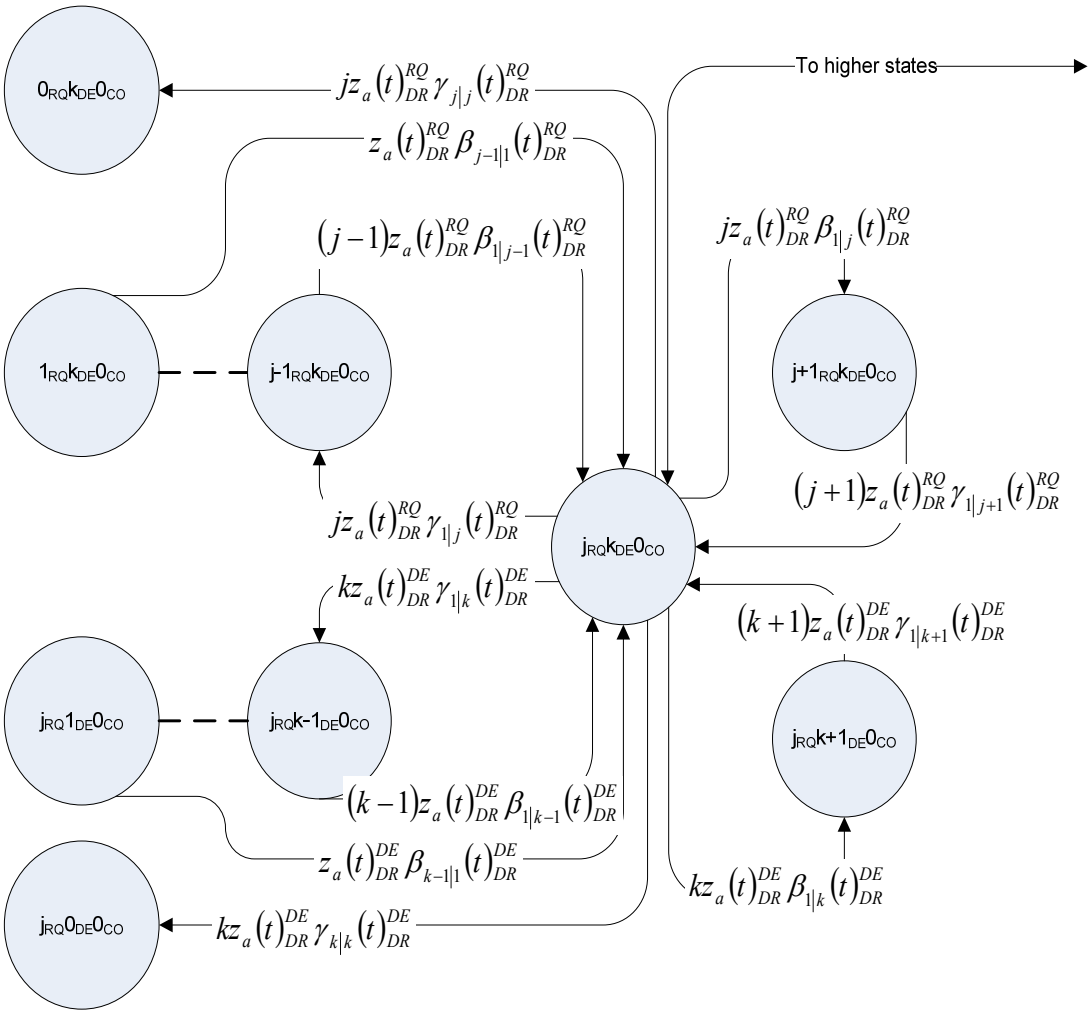


Figure 11 Inside the DR Phase

The Kolmogorov model represented inside the Markov chain state for the DR phase (Figure 11) consists of fault removal components. Of the three types of faults, RQ faults (represented by RQ) and DE faults (represented by DE) will be modified due to applicable transitions. The count for the other fault type does not vary as a result of CO faults not being created in the lifecycle as yet. Since this phase has two active fault types the transitions may occur for either or both types of faults. This phase is the primary phase at which design faults are removed through design reviews.

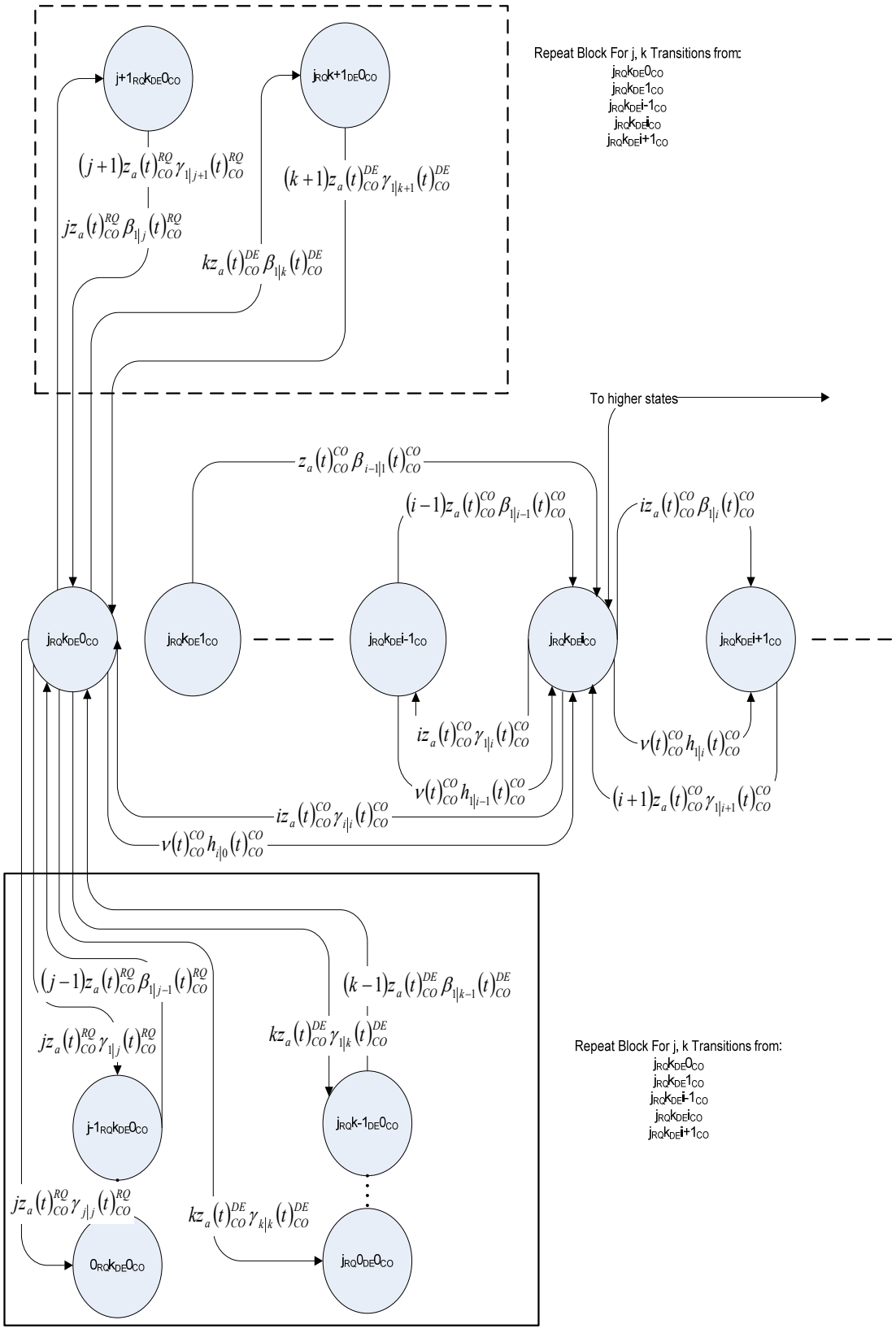


Figure 12 Inside CO Phase

The Kolmogorov model represented inside the Markov chain state for the CO phase (Figure 12) consists of fault introduction and fault removal components. All three types of faults, RQ faults (represented by RQ), DE faults (represented by DE) and CO faults (represented by CO) will be modified due to applicable transitions. Since this phase has three active fault types the transitions may occur for either or all types of faults.

The remaining phases are captured in Figure 13. They are combined into one model since they have the same fault removal transitions. The Kolmogorov model representing the inside of the Markov chain state for these phases (Figure 13) contains three types of faults, RQ faults (represented by RQ), DE faults (represented by DE) and CO faults (represented by CO). Each phase; CI, UT, IgT, ST and AT is a fault removal phase. CI is the primary fault removal phase for CO faults through code inspection. The phases UT, IgT, ST and AT are testing phases and detect each type of fault. Fault detection is most efficient when a detection phase occurs in close proximity to where the fault was generated. The testing phases, which occur late in the lifecycle, require expensive rework to fix faults. Thus, these testing phases do not detect faults, of any of the types, as inexpensively as a review or inspection phase earlier in the lifecycle might.

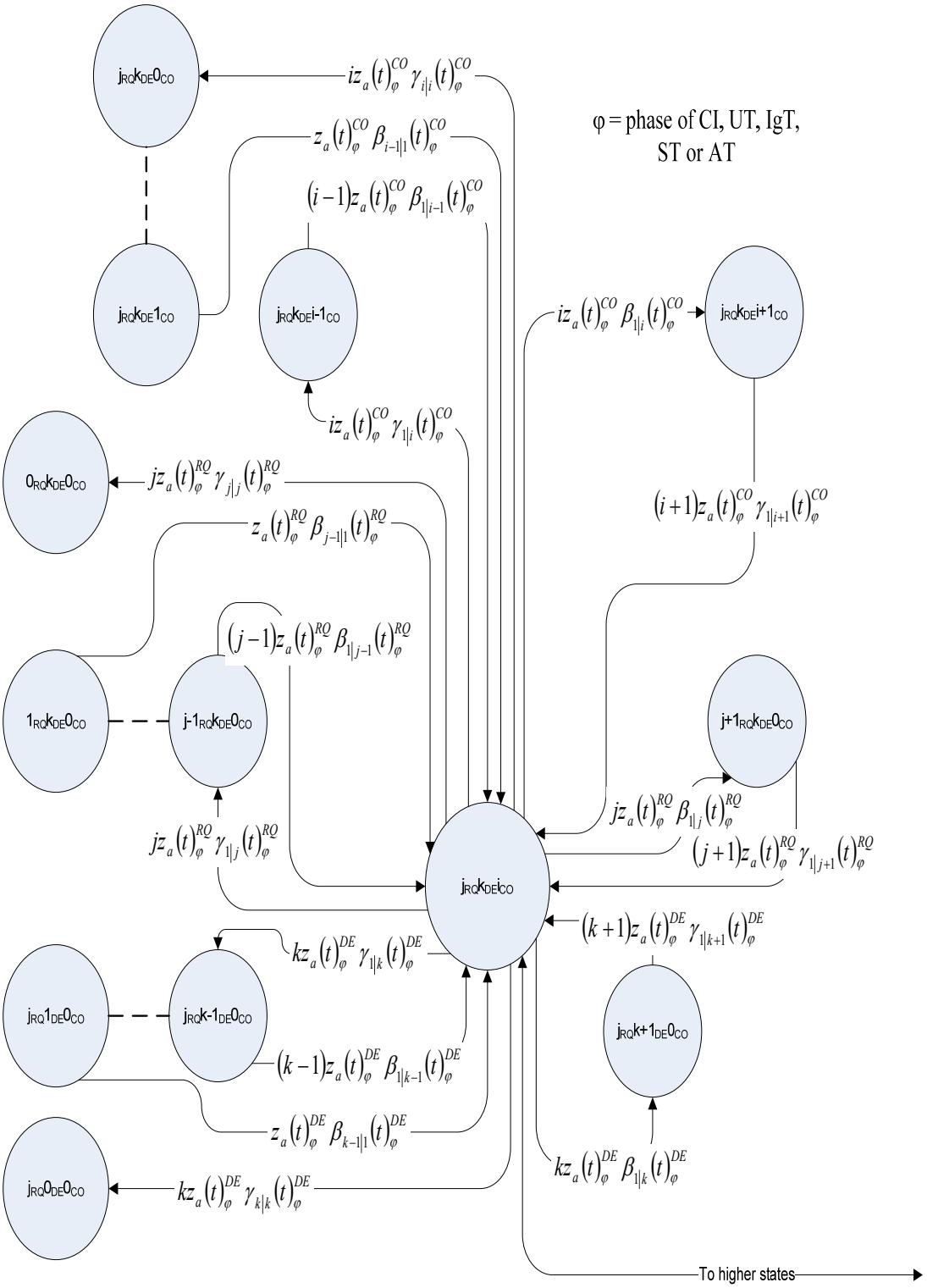


Figure 13 Inside CI, UT, IgT, ST and AT Phases

2.3 *The Waterfall Project*

The waterfall project is a device that delivers a programmable current source. The device's goal is to understand the effects of applying an electrical current to various surfaces. A research device was constructed that allows for constant electrical current of different characteristics to be delivered to eight different channels. The application contains an embedded real-time controller, a Windows user-interface, and a database. The software written for the application stores test data, controls the electronic components of the system and provides sequencing for the running of tests.

The embedded software was developed for a Texas Instruments© MSP430F169 microprocessor along with a Cypress© CY7C68013A56PVC microprocessor and compiled using a Keil© µVision 3 ANSI C compiler. The Windows application was developed in the Microsoft Visual Studio© 2005 environment using the Microsoft Visual C# language. The database was a SQL-based Microsoft SQL Server© database.

2.3.1 *Waterfall Project Function Point Analysis*

Function points are an effective unit-of-work measure of software effort [9]. They meet the acceptable criteria of being a normalized metric that can be used consistently and with an acceptable degree of accuracy [9]. Function point analysis (FPA) measures the functionality being delivered to the user regardless of the technology employed to create the function. This characteristic lends FPA for use in many different application areas. These include Graphical User Interfaces, control applications, embedded applications, data storage applications and many others.

The function point method accounts for the characteristics of the system. The method accounts for data entering a system, external inputs (EI), data leaving a system, external outputs (EO), and external inquiries (EQ), data that is manufactured and stored within the system, internal logic files (ILF), and data that is maintained outside the system but necessary, external interface files (EIF). Other elements of FPA are data element types (DET) and record element types (RET). A record element type is a user recognizable subgroup of data elements within an ILF or EIF. A data element type is a unique user recognizable, non recursive, field. After the components have been classified as one of the five major components (EI, EO, EQ, ILF or EIF), a ranking of low (simple), average or high (complex) is assigned. For transactions (EI, EO, EQ) the ranking is based upon the number of files updated or referenced (FTR) and the number of data element types (DET). For both ILF and EIF files the ranking is based upon record element types (RET) and data element types (DET) [9].

The function point count, for the project, was obtained using the function point counting rules as set forth by the International Function Point Users Group (IFPUG) in the Counting Practices Manual Version 4.1 and referenced by Garmus [9]. Table 4 provides an overview of the function point count of the Waterfall project.

Table 4 Waterfall Project – Unadjusted Function Point Table

Business Function	Number	Complexity	Factor	Line Total	Group Total
EXTERNAL OUTPUTS (EO)	1	Simple	* 4 =	4	
		Average	* 5 =		
		Complex	* 7 =		
TOTAL:					4
EXTERNAL INPUTS (EI)	2	Simple	* 3 =	6	
		Average	* 4 =		
	1	Complex	* 6 =	6	
TOTAL:					12
EXTERNAL INQUIRIES (EQ)	2	Simple	* 3 =	6	
		Average	* 4 =		
		Complex	* 6 =		
TOTAL:					6
INTERNAL LOGICAL FILES (ILF)	5	Simple	* 7 =	35	
		Average	* 10 =		
		Complex	* 15 =		
TOTAL:					35
EXTERNAL INTERFACES (EIF)	0	Simple	* 5 =		
		Average	* 7 =		
		Complex	* 10 =		
TOTAL:					0
UNADJUSTED FUNCTION POINTS **(UFP)**				=	57

After an unadjusted function point count has been obtained for the application the characteristics of the system are accounted for. These are called General System

Characteristics (GSC). The GSC's as identified by Garmis [9], as also stated by IFPUG, and is shown as 14 system characteristics, each of which is rated in terms of its degree of influence (DI) on a scale of 0 to 5:

0 Not present, or no influence
1 Incidental influence
2 Moderate influence
3 Average influence
4 Significant influence
5 Strong influence throughout

The GSC's are:

1. Data communications
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. Online data entry
7. End user efficiency
8. Online update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

The terms Total Degree of Influence (TDI), Value Adjustment Factor (VAF) and Adjusted Function Point Count (FP) are used and consistent with the IFPUG

approach. TDI indicates the total sum of the GSC component. The TDI can vary from 0 (when all GSCs are low) to 70 (when all GSCs are high). VAF indicates the adjustment parameter applied to the unadjusted function point count. The VAF can vary in range from 0.65 (when all GSCs are low) to 1.35 (when all GSCs are high). The VAF can vary from 0.65 to 1.35, so the VAF can affect the final FP value by +/- 35%. FP is the value of the function point count after applying adjustments. The final function point value for the Waterfall project is found through the following equations.

Table 5 General System Characteristics – Waterfall Project

General System Characteristic	Degree of Influence (0 to 5)
Data communications	3
Distributed data processing	2
Performance	5
Heavily used configuration	3
Transaction rate	4
Online data entry	5
End user efficiency	3
Online update	3
Complex processing	2
Reusability	2
Installation ease	2
Operational ease	3
Multiple sites	0
Facilitate change	2
Total	39

TDI = 3 + 2 + 5 + 3 + 4 + 5 + 3 + 3 + 2 + 2 + 2 + 3 + 0 + 2 = 39 (2.34)

VAF = (TDI * 0.01) + 0.65 = (39 * 0.01) + 0.65 = 1.04 (2.35)

FP = UFP * VAF = 57 * 1.04 = 59.28 (2.36)

2.3.2 Waterfall Project Parameter Definitions and Derivations

The parameters used for the model in the Waterfall project are calculated from industry data. These parameters are used to calculate the Fault Introduction phase values and the Fault Removal phase values. Since these parameters are based on industry obtained metrics there is an uncertainty when applied to applications due to variations in industry development environments. These values are matched as closely as possible to the industry metrics which align most closely with the type, size, complexity, nature and environment of the application.

The parameters used in the model for the fault introduction phases are considered first. The equation for the Fault Introduction phase is:

$$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi} = \frac{DP_\varphi \bullet fd_{j,\varphi}}{\overline{t_{FP,\varphi}}} \bullet F_{j,\varphi}^{1-2SLI_\varphi} \quad (2.37)$$

The parameter, $F_{j,\varphi}$, is a calculated value from equation 2.14. From equation 2.14 and equations 2.11 to 2.13, boundary conditions (maximum and minimum) values are needed.

Table 6 Defect Potential (DP_{φ} * $fd_{j,\varphi}$) per Function Point per Phase

Defect Origins	Average Defect Potential [15]	The Upper Bound of the Defect Potential [16]	The Lower Bound of the Defect Potential [17]
Requirements	1.00	1.50	0.40
Design	1.25	2.20	0.60
Coding	1.75	2.50	1.00
Documents	0.60	1.00	0.40
Bad Fixes	0.40	0.80	0.10
Total	5.00	8.00	2.50

In Table 6, based on Jones [15] [16] [17] the average defect potential, upper bound of defect potential and lower bound of defect potential per function point per phase is shown. Next, we find the value for the mean effort per function point per lifecycle phase, ($t_{FP,\varphi}$) in Table 7.

Table 7 Mean Effort per Function Point per Lifecycle Phase φ, in Staff Hours

Phase, φ	Max	Mode	Min	Mean *	Notes
RQ	2.64	0.75	0.38	1.00	
RR	1.76	0.59	0.33	0.74	Requirements Reviews
DE	9.90	2.33	1.12	3.39	Arch.+Proj.Plans+Init.Des.+Detailed Des.
DR	1.76	0.59	0.33	0.74	
CO	8.80	2.64	0.66	3.34	
CI	1.76	0.88	0.44	0.95	
UT	1.89	0.88	0.33	0.96	Unit Testing
IgT	1.76	0.75	0.33	0.85	Integration Testing
ST	1.32	0.66	0.26	0.70	System Testing
AT	1.76	0.38	0.22	0.58	Acceptance Testing

*** Note [12]:**

$$Mean = \frac{1}{6}\left(Min + \left(4 * Mode\right) + Max\right) \tag{2.38}$$

Next, the boundary conditions for (DP_{φ}), ($fd_{j,\varphi}$) and ($t_{FP,\varphi}$) are tabulated from the information above.

Table 8 Boundary Information for $DP_{\varphi} \bullet fd_{j,\varphi}$ and $\bar{t}_{FP,\varphi}$

	Requirements			Design			Coding		
	Max	**Mean**	**Min**	**Max**	**Mean**	**Min**	**Max**	**Mean**	**Min**
$DP_{\varphi} \bullet fd_{j,\varphi}$	1.50	1.00	0.40	2.20	1.25	0.60	2.50	1.75	1.00
$\bar{t}_{FP,\varphi}$	2.64	1.00	0.38	9.90	3.39	1.12	8.80	3.34	0.66

Normally, there are enough reasons to believe that the defect potential will become smaller if more effort is spent on the development process. Thus, the maximum defect potential is corresponding to the minimum effort ($t_{fp,\varphi\ MIN}$) and the minimum defect potential is corresponding to the maximum effort ($t_{fp,\varphi\ MAX}$). Therefore [23], the upper bound of the $\{\nu(t)\mu_H(t)\}_{j,\varphi}$ can be obtained by using the maximum defect potential divided by the minimum development effort. The lower bound of the $\{\nu(t)\mu_H(t)\}_{j,\varphi}$ can be obtained by using the minimum defect potential divided by the maximum development effort.

Table 9 Boundary Information for $\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi}$

	Requirements Phase			Design Phase			Coding Phase		
	Max	**Mean**	**Min**	**Max**	**Mean**	**Min**	**Max**	**Mean**	**Min**
$\{\nu(t) \bullet \mu_H(t)\}_{j,\varphi}$	1.50/ 0.38	1.00/ 1.00	0.40/ 2.64	2.20/ 1.12	1.25/ 3.39	0.60/ 9.90	2.50/ 0.66	1.75/ 3.34	1.00/ 8.80
	3.95	1.00	0.15	1.96	0.37	0.061	3.79	0.524	0.114

Thus, the value of $F_{j,\varphi}$ for each development phase can be obtained from Equation 2.39 and shown in Table 10.

$$F_{\varphi} = \sqrt{\frac{(\nu(t) \bullet \mu_H(t))_{\max}}{(\nu(t) \bullet \mu_H(t))_{\min}}} \qquad (2.39)$$

Table 10 Values of F for the Fault Introduction Phases

	RQ	DE	CO
F	$\sqrt{\frac{3.95}{0.15}}$	$\sqrt{\frac{1.96}{0.061}}$	$\sqrt{\frac{3.79}{0.114}}$
	5.13	5.67	5.76

The Defect Potential (DP_{φ}), parameter is calculated using a weighted mean approach. The defect potential is related to the function point count of the application, as more effort is applied to develop an application there is more potential of defects. In this research, data for defect potential was obtained from two sources [6] [21], one from 1996 and one from 2008. These two values are used to calculate the weighted mean value of defect potential. The function point count value as developed in equation 2.36 is 59.28. The application is categorized as being "Systems & Embedded Software". The function point count data for defect potential is grouped in powers of ten, thus, this application fits in the range, $10 < 59.28 < 100$ function point counts.

The function point and defect potential pairs used for the calculation are grouped by function point count in powers of ten. This characteristic leads to performing a logarithmic interpolation. Then the weighted mean is applied to account for the varying data from the years in which the industry data is obtained. Thus, logarithmic interpolation is used to account for the function point divisions in powers of ten and the weighted mean is used to account for the spread in industry data.

From the data in Table 7 of [21] for the 2008 defect potential metric:

For Systems & Embedded Projects:

Function Points = 10 Defect Potential = 2.50

Function Points = 100 Defect Potential = 3.00

The logarithmic interpolation method is used to find the value of DP at a function point count of 59.28.

$$x = x_1 + \frac{x_2 - x_1}{\log_{10}(y_2) - \log_{10}(y_1)}\left(\log_{10} y(x) - \log_{10}(y_1)\right) \quad (2.40)$$

Interpolating this defect potential range yields the DP for the 2008 metric:

$$DP(2008) = 2.50 + \frac{3.00 - 2.50}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(59.28) - \log_{10}(10)\right) = 2.89 \quad (2.41)$$

For the 1996 defect potential metric the following table is used.

Table 11 Fault Potential per Function Point, DP

Function Points	End User	MIS	Outsourced	Commercial	Systems	Military	Average
1	1	1	1	1	1	1	1.00
10	2.5	2	2	2.5	3	3.25	2.54
100	3.5	4	3.5	4	5	5.5	4.25
1,000	n/a	5	4.5	5	6	6.75	4.54
10,000	n/a	6	5.5	6	7	7.5	5.33
100,000	n/a	7.25	6.5	7.5	8	8.5	6.29

$$DP(1996) = 3 + \frac{5 - 3}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(59.28) - \log_{10}(10)\right) = 4.55 \quad (2.42)$$

Then, the weighted mean value for defect potential is obtained from values for DP(1996) and DP(2008). The application was completed in the year 2006.

Calculation for Weighted Mean of Defect Potential (DP):

Defect Potential Range = 4.55 – 2.89 = 1.66

Year Range = 2008 – 1996 = 12

$$\frac{1.66}{12} = \frac{x}{2006 - 1996}$$

$$x = 1.38$$

Therefore, the value for DP_{φ} is:

DP(Waterfall Project) = (4.55 – 1.38) = 3.17

Next, the value for the fraction of faults that originated in a phase φ, ($fd_{j,\varphi}$), is determined. This parameter is necessary to use in the computation of the model for Fault Introduction. The ($fd_{j,\varphi}$), part of $DP_{\varphi} * fd_{j,\varphi}$, found for Table 8 was used in the calculation of $F_{j,\varphi}$. The following table supplies the fraction of faults per phase categorized by phase for application type, the waterfall project has been categorized as a Systems application.

Table 12 Software Defect Origin Percent by Industry Segment per Phase

Phase, φ	$Fd_{j,\varphi}$						
	End User	**MIS**	**Outsourced**	**Commercial**	**Systems**	**Military**	**Average**
RQ	0.0	0.15	0.20	0.10	0.10	0.20	0.1250
DE	0.15	0.30	0.25	0.30	0.25	0.20	0.2417
CO	0.55	0.35	0.35	0.30	0.40	0.35	0.3833
User Document	0.10	0.10	0.10	0.20	0.15	0.15	0.1333
Bad Fix	0.20	0.10	0.10	0.10	0.10	0.10	0.1167

The SLI value used with the $F_{j,\varphi}$ parameter is computed as discussed in previous sections. An expert opinion elicitation was performed to assess the quality of the influencing factors of Table 2 for the Waterfall project. The results of the elicitation are given in Appendix B.

The Fault Removal activities for Fault Introduction phases follow equation 2.22 for the removal rate. This equation uses as parameters the time spent in a phase, t, and the Defect Removal Efficiency (DRE) for that phase. In Fault Introduction phases the developer performs the defect removal activity of desk checking. This is when the developer takes time away from development activities to perform an inspection of the work in progress. In Fault Removal phases, equation 2.32 determines the removal rate. This equation uses the parameters $t_{FP,\varphi}$ and DRE_{φ}. In Fault Removal phases the reviewer, inspector, or tester will be actively pursuing defects. Tables in Appendix A give the DRE for various removal activities which correspond to the project phases.

2.3.3 Waterfall Project Data

The data collected for the Waterfall project consists of actual staff hours and actual observed faults in selected phases. The SLI is calculated from expert opinion.

The actual staff hour data collection was a combination of time recording and observation. A comparison can be made to the estimate of staff hours obtained by calculation of industry data. The actual staff hour data comes from observations recorded on the Waterfall project. The estimated staff hour data comes from industry data. From Table 7 the mean effort in staff hours, given by industry, is gathered in Table 13. The estimated staff hours per phase is calculated from the product of the mean effort per function point in staff hours, t_{FP}, and the function point count in equation 2.36.

Table 13 Mean Effort In Staff Hours Per Function Point per Phase ($t_{FP,\varphi}$)

Activity	RQ	RR	DE	DR	CO	CI	UT
Effort	1.00	0.74	3.39	0.74	3.34	0.95	0.96
Activity	**IgT**	**ST**	**AT**				
Effort	0.85	0.70	0.58				

Table 14 Estimated Staff Hours per Phase from Function Points

Activity	Staff Hours
RQ	1.00*59.28=59.28
RR	0.74*59.28=43.87
DE	3.39*59.28=200.96
DR	0.74*59.28=43.87
CO	3.34*59.28=198.00
CI	0.95*59.28=56.32
UT	0.96*59.28=56.91
IgT	0.85*59.28=50.39
ST	0.70*59.28=41.50
AT	0.58*59.28=34.38

A comparison to the actual data reveals variations per phase.

Table 15 Lifecycle Phase In Staff Hours (Estimated and Actual)

Activity	Staff Hours (From Function Points)	Staff Hours (From Actual Hours)
RQ	59.28	80
RR	43.87	48
DE	200.96	157.5
DR	43.87	67.5
CO	198.00	282.6
CI	56.32	16
UT	56.91	94.2
IgT	50.39	94.2
ST	41.50	50
AT	34.38	44
Total	785.48	934

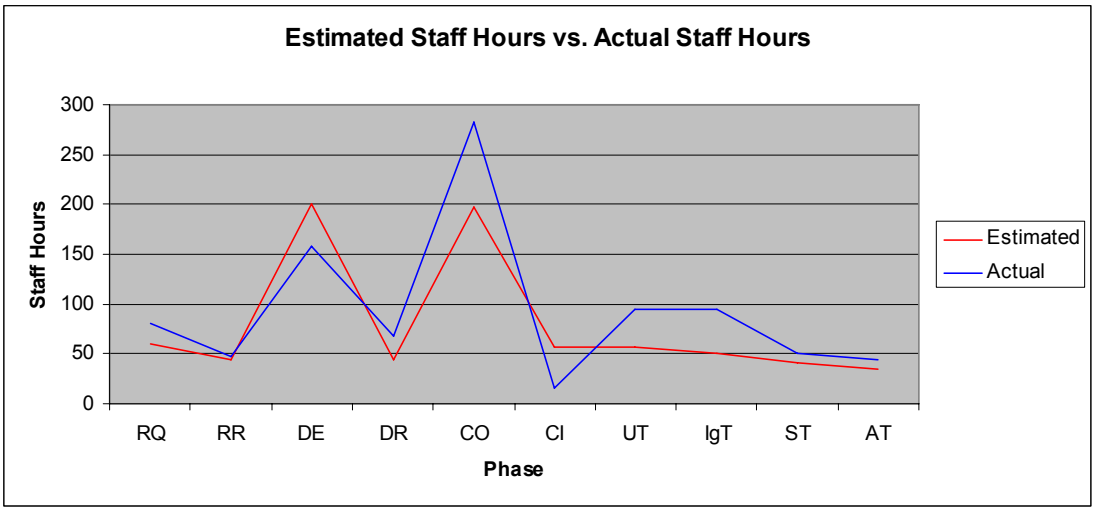


Figure 14 Estimated vs. Actual Staff Hours – Waterfall Project

An observation obtained in analyzing the staff hours data is the under-estimate obtained from the function point count calculation of staff hours for the overall project effort. There appears to be a trend that more effort was expended in the review and testing phases than given by the standard industry metrics for those phases. The code inspection phase did not contribute to this under-estimate however. A hypothesis for the time spent in actual testing could be revealed in the 70% more effort expended in the coding (CO) phase than industry data would estimate. This corresponds to the 65% more effort expended in the testing (UT, IgT, ST and AT) phases than predicted by the function point count estimate. The hypothesis would follow that since much more effort was expended in coding a similar effort would be required in testing.

Faults were observed and recorded in three phases, Requirements Review (RR), Design Review (DR) and System Test (ST). For the RR phase, the faults recorded were requirements faults, as they were the only type available at that time. For the DR phase, the faults recorded were design faults, as they were the category of

fault reviewed during the design review process of this project. In the ST phase, all categories of faults; requirements, design and coding faults, were recorded as a composite value.

Table 16 Observed & Repaired Faults Found per Phase (Waterfall)

Phase	Number of Faults Actually Observed & Repaired		
	Category 1. Critical	**Category 2. Major**	**Total**
RQ Faults - Requirements Review (RR)	6	0	6
DE Faults – Design Review (DR)	7	0	7
RQ, DE, CO Faults - System Test (ST)	7	0	7
Total	**20**	**0**	**20**

2.3.4 Waterfall Project Software Reliability Model Results

The reliability model was applied, with the preceding data, to the Waterfall project. The fault predictions of the reliability model are based on the industry-based metrics, the calculated metrics based on the industry-based metrics and expert opinion-based parameters. At this point, the expert-opinion parameter, SLI, is only applied to the Fault Introduction phases. Refer to Appendix B for the calculation of the SLI from expert opinion for the Waterfall project. An enhancement to the model will make the SLI parameter available to Fault Removal phases in Chapter 4. The SLI used is based on expert opinion; an extension to the model will optimize the SLI parameter based on the observable data also in Chapter 4. The staff hours parameter

used in the model is the actual staff hours and not the estimated staff hours. The parameters for this test case (#40) are given in Appendix D.

The output from the model includes the value of the model predicted requirements, design, coding and cumulative faults at each phase in the development cycle.

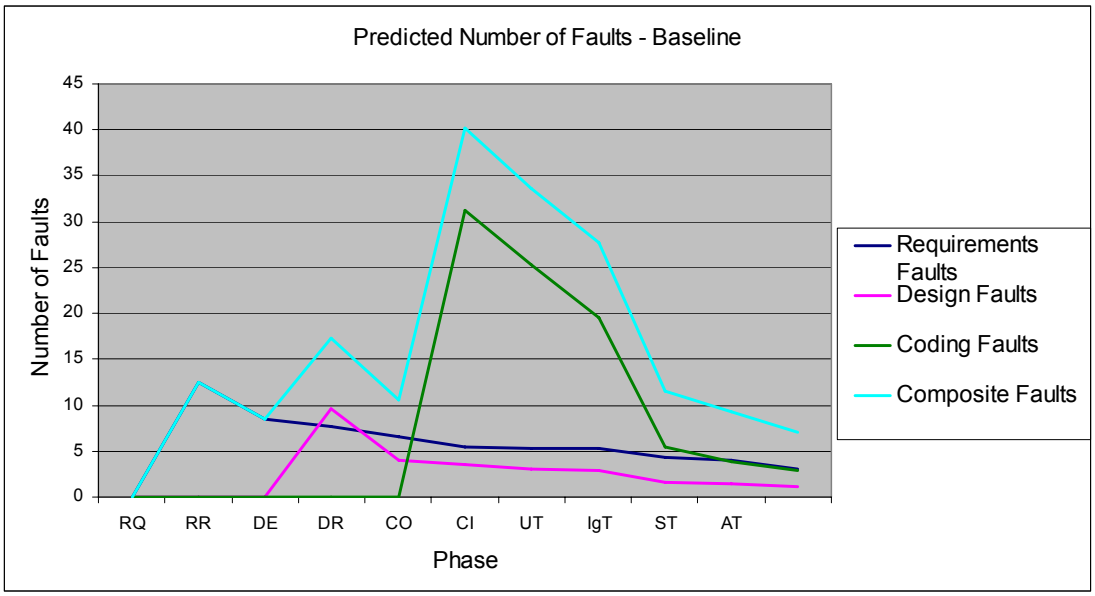


Figure 15 Model Predicted Fault Count – Waterfall Lifecycle

Table 17 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle

Inspection Phase	Fault Type	Expected Faults	Observed & Repaired Faults
RR	RQ Faults	4.05	6
DR	DE Faults	5.54	7
ST	Cumulative Faults	2.26	7
Total		11.85	20
Percent Error (Expected vs. Observed Faults)		40.75	

The reliability model, with the parameters as above, predicts that 7.06 faults could be expected in the system at the end of development. This number will change as enhancements and optimizations are applied to the model in Chapter 4.

The "Percent Error" measure is defined as the per cent of absolute value of: (Expected Observed Faults – Actual Observed Faults) / Actual Observed Faults. This measure is applied to the total expected observed and actual observed faults. The percent error in the measurement illustrates the measure of distance of the total expected observed faults from the model predictions to the actual observed faults. The percent error is valuable in determining performance of the model as a means of predicting observed faults.

Table 17 indicates that the model parameter values predicted lower fault values than observed and also indicates optimizations are necessary. These enhancements and optimizations will be performed in Chapter 4. An analysis of the results of the model compared to the model with enhancements occurs in Chapter 4.

Chapter 3: Development of the Reliability Model for Feature Driven Development (FDD) – An Agile Approach

3.1 FDD Lifecycle Description

An implementation of the Agile software development process called Feature Driven Development (FDD) is described in this chapter. An overview of the principles behind FDD was given in section 1.4.3.

The Agile software development process is typically used by adopting a subset of Agile principles. One subset of the Agile principles is the FDD process. This process has been adopted for use in this research. The FDD process [5]:

- Is highly iterative
- Emphasizes quality at each step
- Delivers frequent, tangible, working results
- Provides accurate and meaningful progress and status information
- Is liked by clients, managers and developers

The FDD process is best applied when a project is complex yet can be broken down into smaller components. Other aspects of when FDD is best is when a team of developers who have good communication lines are available (likely co-located), when regular communication takes place among the development team (short and frequent meetings called scrums), when the users of the developed system are active participants in the development and when frequent, tangible working results are desired by the user. These types of projects have been documented in most business areas.

The lifecycle would likely not be used in a mission-critical, highly regulated project development, although this is often disputed by Agile enthusiasts.

The advantages of an FDD process are in its ability to provide an approach to solving complex projects, the ease of team communications, the user involvement and frequent product deliveries. The disadvantages of FDD are centered on what this process does not do. It does not provide the team or the user with a big documentation effort "up-front". It allows more opportunity for changes to occur during the process. Many software metrics for this type of development do not exist and it may be more difficult to collect metrics from this type of process.

The FDD lifecycle used in this research consists of a Requirements (RQ), Requirements Review (RR), System Design (SD), System Design Review (SDR), System Model (SM), System Model Review (SMR), Feature List (FL), Feature List Review (FLR) and a Feature List Planning (FLP) phase. The FDD lifecycle also contains an iterative component. The iterative phases in the implementation are Feature Design (DE), Feature Design Review (DR), Feature Coding (CO), Feature Code Inspection (CI), Feature Unit Test (UT), Feature Integration Test (IgT), System Test (ST) and Acceptance Test (AT).

The individual phases have the same behavior as in their Waterfall lifecycle counterparts described in section 2.2. The RQ and RR phases collect the requirements for the system and review those requirements. The FDD unique phases of SD, SDR, SM, SMR, FL, FLR and FLP each have a behavior in the process. The SD, SDR, SM and SMR phases correspond to the "Develop an Overall Model" activity of the characteristic FDD lifecycle, as shown in Figure 2. These phases

provide an initial design and architecture, in the form of a model, of the system as a whole along with corresponding reviews of this design. These phases are design phases. The FL and FLR phases correspond to the "Build Features List" activity of Figure 2. These phases perform a further refining of the requirements by creating functional categories, thus are thought of as a requirements phase. In this research, the functional categories were broken down into two further refinements. Major function categories were described as Feature Areas, the subcategories under Feature Areas were Feature Activities, and these activities were broken down to a list of Features. The Feature List was then reviewed during the FLR phase. The FLP phase produces a full development list from the Feature List. This phase corresponds to the "Planning" activity of the FDD lifecycle in Figure 2. This phase is not reviewed. The phases of DE, DR, CO, CI, UT, IgT, ST and AT behave as a set of iterations of a Waterfall lifecycle. The Feature List along with the Feature Planning produces a set of features for each iteration of the phases of DE, DR, CO, CI, UT, IgT, ST and AT to implement. In this research the phases of DE, DR, CO, CI, UT and IgT produce an implementation based on the current Feature List while the phases of ST and AT perform regression testing of the current iteration and all of development done in previous iterations. The number of iterations is determined by the length of the Feature List, the number of features per iteration, the effort per feature and the desired duration of each iteration.

The phases of RQ, SD, SM, FL, DE and CO can create faults. DE and CO may iterate a number of times and create a set of faults per iteration. Each of the previous fault introduction phases is followed by a review, inspection or testing

phase. Documentation is created for the RQ, SD, SM, FL, DE and CO phases. The RR, SDR, SMR, FLR, DR, CI, UT, IgT, ST and AT phases remove faults from the lifecycle.

In this research, for the FDD lifecycle project, documentation was kept for faults observed in the RR (requirements faults), SDR (design faults), SMR (design faults), FLR (requirements faults), DR (design faults), one CI phase (coding faults) and ST (requirements, design and coding faults) phases.

Table 18 Feature Driven Development (FDD) Lifecycle Phase

Phase	Acronym	Architecture	Preliminary Mapping to Known Lifecycle Phases
Requirements	**RQ**	System	Requirements (Functional)
Requirements Review	**RR**	System	Requirements Reviews (Functional)
System Design	**SD**	System	Initial Design (Functional)
System Design Review	**SDR**	System	Design Reviews (Functional)
System Model	**SM**	Software	Architecture
System Model Review	**SMR**	Software	Design Reviews
Feature List	**FL**	Software	Requirements
Feature List Review	**FLR**	Software	Requirements Reviews
Feature List Plan	**FLP**	Software	Project Plans
Feature DE	**DE**	Software	Detailed Design
Feature DR	**DR**	Software	Design Review
Feature CO	**CO**	Software	Coding
Feature CI	**CI**	Software	Code Inspection
Feature UT	**UT**	Software	Unit Testing
Feature IgT	**IgT**	Software	Integration Testing
System Test	**ST**	Software	System Testing
System AT	**AT**	Software	Acceptance Testing

The adaptation of this lifecycle to the software reliability model used in this research is one of the contributions made by this thesis. The Waterfall lifecycle phases contained within this process made the mapping to the FDD phases possible.

Research was performed to accurately place the phases into correct categories; Table 18 displays the mapping of FDD phases to known Waterfall lifecycle phases. The research identified lifecycle phases, determined the metrics available for these phases, assessed the suitability of a known lifecycle phase to an FDD phase and performed the mapping. The goal of the mapping was to categorize the FDD phases to a known lifecycle phase with available metrics. The iterative nature of the FDD lifecycle was also accounted for in the mapping as it pertains to function point counting, this is discussed in section 3.2.1.

3.1.1 FDD Lifecycle Phases

Each lifecycle phase in the FDD software development process, as implemented in this research and illustrated in Figure 16 and Figure 17, has an input, output, next phase and possible fault feedback path for fault removal phases. The phases may create documentation, review documentation, introduce faults, or remove faults. Faults are introduced during phases which design or construct software; these defects are known as development faults. These faults are categorized as requirements, design or coding defects. Faults may also be introduced during attempts to remove a known defect; these are known as debugging faults. Faults are removed by the processes of reviews, inspections and tests. These phases are known as fault removal phases. This FDD implementation is one possible implementation of the phases contained in the literature concerning FDD software development. In the FDD lifecycle illustrated in the figures, each phase displays corresponding Waterfall phases, if they exist, the FDD phase and the engineering discipline associated with that phase.

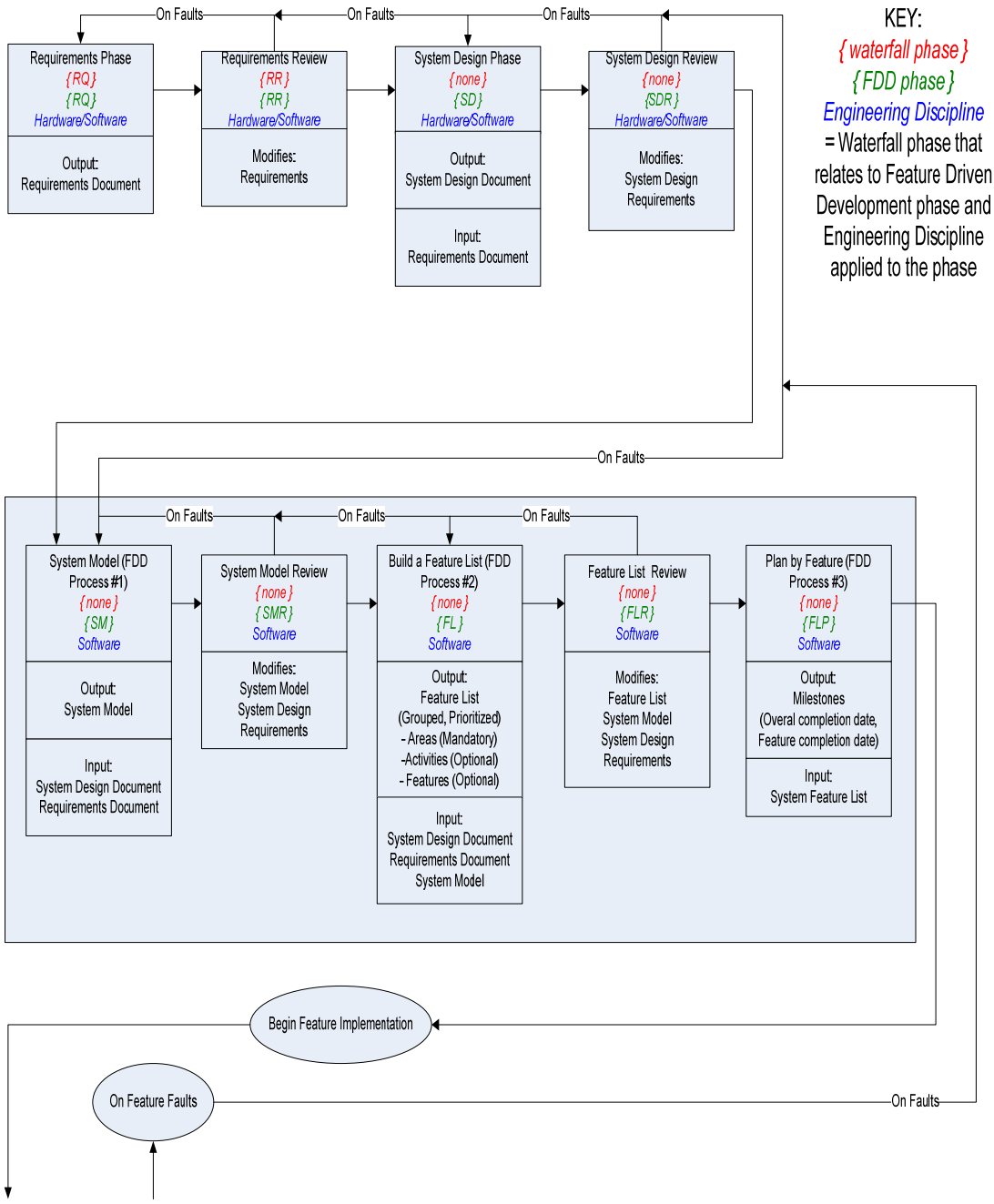


Figure 16 Feature Driven Software Development Lifecycle Phases (Pre-Iteration)

Figure 16 continues to and from Figure 17. The iterative feature implementation is depicted in Figure 17. The flow of development is shown in the

outgoing “Begin Feature Implementation” arrow and the flow of fault data to system design and requirements phases is indicated in the “On Feature Faults” arrow.

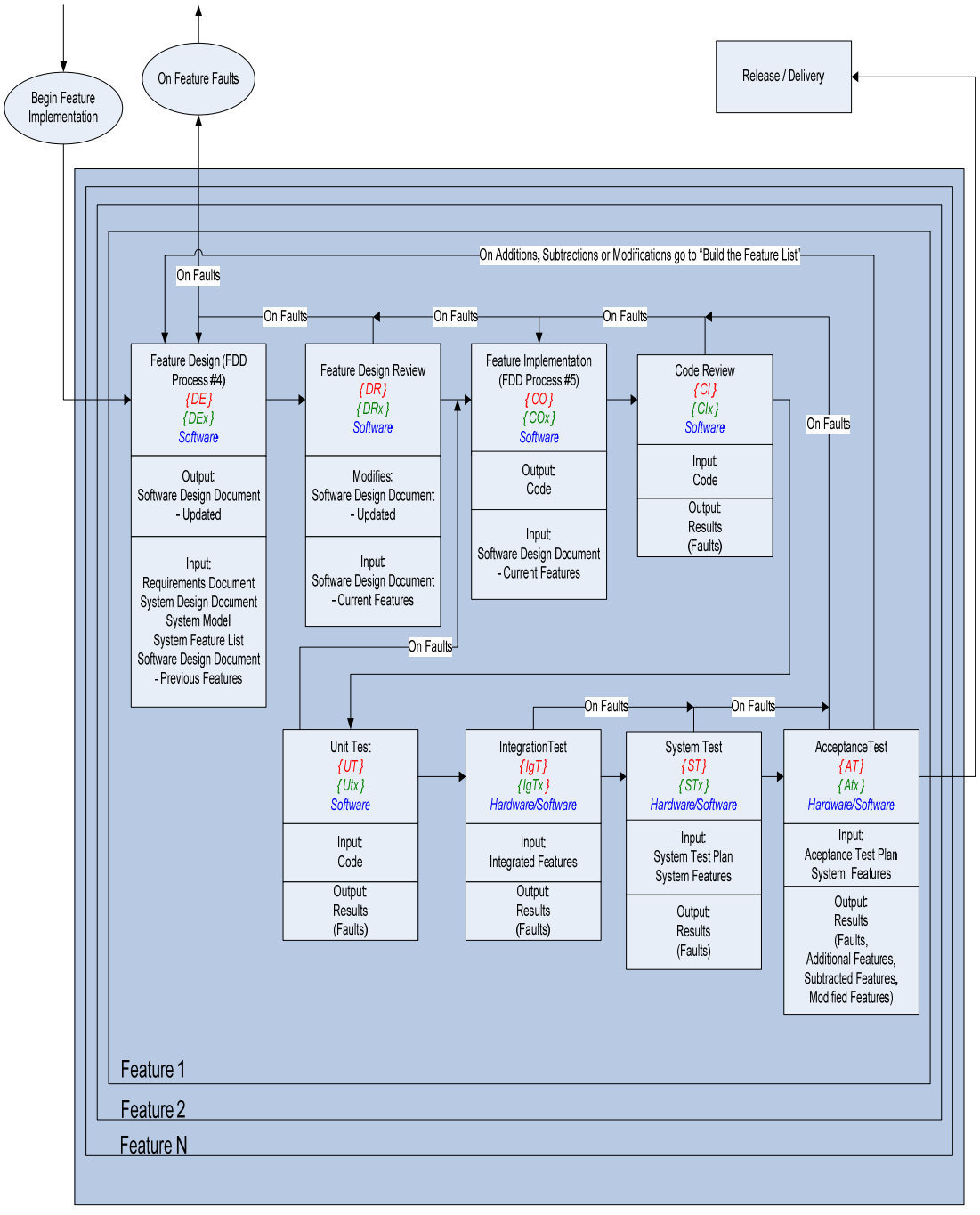


Figure 17 Feature Driven Development Software Lifecycle Phases (Iterations)

Figure 17 illustrates the flow of feature implementation, from the pre-iterative phases of the lifecycle, and the flow of fault data back to system phases for repair. Each feature implements a full waterfall process with regression testing performed in ST and AT.

3.1.2 Markov Model of the FDD Lifecycle

The FDD lifecycle is an NHMBDWI process. Because it has the characteristics of an iterative waterfall lifecycle it can be represented by the NHMBDWI process, as is the Waterfall lifecycle. Section 2.1 gives the theory behind the development of the NHMBDWI process from a model of Kolmogorov forward equations. The state transition diagram for development errors, bad repairs and good repairs in Figure 5 represents the same behavior that occurs inside the phases, fault introduction and removal, of the FDD lifecycle. Transitions between phases of the FDD lifecycle can be modeled as a Markov model. The Dirac Delta (δ) function represents the conditions present for state transitions, as in the Waterfall lifecycle.

The Markov state transition diagram, in Figure 18 displays the waterfall-like appearance along with the iterative aspect of the DE, DR, CO, CI, UT, IgT, ST and AT phases. The variables in the diagram are:

$N_{FEATUREfp,\varphi,k}(j)$	number of function points in a feature for phase φ in iteration k
k	iteration number, updated upon iteration completion
j	feature counter, set at the start of an iteration loop and updated upon iteration completion

NTotalFeatures	total number of features in a project
NFeaturesComplete	features completed in the development
t	time spent in a phase
$t_{FP,\varphi}$	time per function point for phase φ
N_{FP}	number of project function points

The Heaviside step function, H, is used and has the value of zero for a negative argument and one for a positive argument.

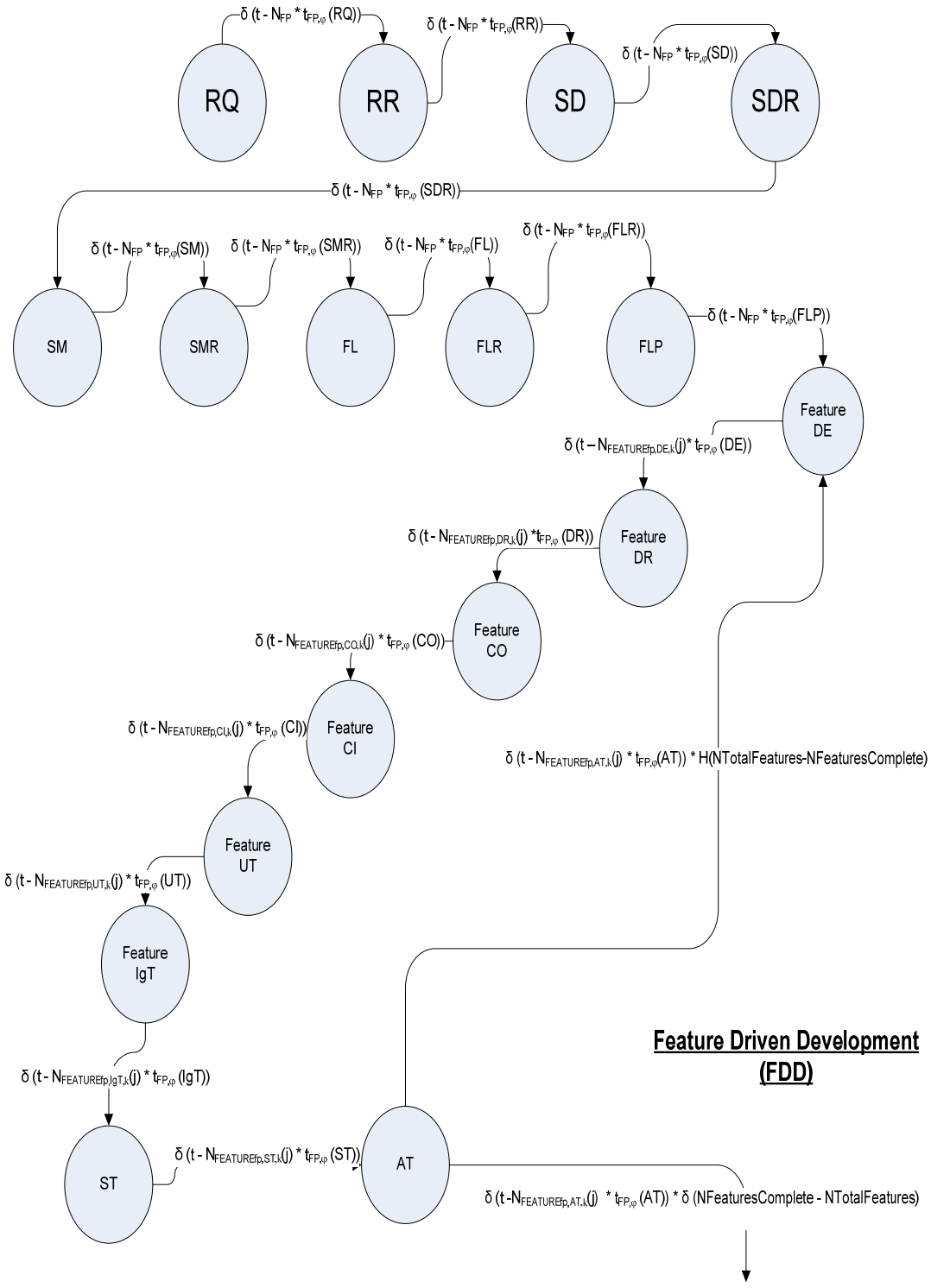


Figure 18 FDD Lifecycle Markov Model

As in the Waterfall lifecycle, the FDD Markov model of Figure 18 can be decomposed into the Kologorov model representing fault introduction and removal occurring in that state. The interiors of the state transition diagram's states would correspond to those found in the Waterfall project.

As in the Waterfall process, the effect of development errors and debugging errors (bad repairs and non-repairs) along with the good repairs achieved are accounted for inside the phases in the Markov model. Figure 8 (RQ), Figure 9 (RR), Figure 10 (DE), Figure 11 (DR), Figure 12 (CO) and Figure 13 (CI, UT, IgT, ST, AT) show the accounting for fault processing of their states, in the Waterfall lifecycle. This representation is equivalent to the representation for the same states in the FDD lifecycle. The FDD lifecycle states SD, SDR, SM, SMR, FL, FLR and FLP have the same characteristic behavior of fault introduction and removal as the states represented in the figures, thus are not represented here.

3.2 The FDD Project

The FDD project under study is composed of a series of applications to control a networked audio/visual system. The applications provide a user interface to the system, an administration interface, control logic for business rules and a data store. The implementing technologies for the system of applications include Microsoft Windows, embedded microcontrollers, Microsoft Windows .NET Framework and SQL databases.

The applications provide a system for two way audio and video over a network between a central command center and various remote stations. The stations

provide for various monitoring functionalities. Various signaling, messaging and logging also are performed in this system.

The Windows applications are developed using the Microsoft Visual Studio 2010 development environment along with the Microsoft .NET version 4 framework. The embedded software was developed for a Microchip PIC 18F2520 microcontroller using the CCS Inc. PCWH ANSI C Compiler IDE, PCB, PCM, PCH, Version 4.073. The database is a SQL-based Microsoft SQL Server database.

3.2.1 FDD Project Function Point Analysis

The FDD lifecycle process, of this system, required research of FPA, and consultation with FPA experts, to determine how to adapt the IFPUG function point counting rules to the FDD lifecycle. This adaptation led to the addition of two additional function point counting rules. The research in the function point counting aspect of the FDD lifecycle is a contribution of this thesis.

When considering function point counting, a system or project can be thought of as being one of two types. These are a "development project" or an "enhancement project." A development project is one that provides initial functionality to a user. This could also be the only functionality provided for the life of the project and be the project's final functionality. An enhancement project performs modifications to an existing baseline project. In the FDD lifecycle, the approach to FPA called for the pre-iteration phases and first iteration to be considered a development project. The succeeding iterations are considered as individual enhancement projects. In summary the phases of the FDD project were considered as both a development and an enhancement project and distributed as follows:

Development Project

Pre-Iteration Phases (RQ, RR, SD, SDR, SM, SMR, FL, FLR, FLP)

- These phases use the full System Function Point Count obtained by performing a Development Project count on the documentation developed in these phases for the full system.

Iteration 0 (DE0, DR0, CO0, CI0, UT0, IgT0, ST0, AT0)

- These phases use the Function Point Count obtained by performing a Development Project count on the documentation developed in these phases for the system implementation specified for iteration 0.

Enhancement Project

Iteration 1 <= n <= MaxIterations (DEn, DRn, COn, CIn,UTn, IgTn, STn, ATn)

- These phases use the Function Point Count obtained by performing an Enhancement Project count on the documentation developed in these phases for the system implementation specified for iteration n. The STn and ATn phases perform regression testing of the Development Project count and each Enhancement Project's count up to the current iteration.

The possibility of creating or modifying the function point counting rules, as described in the "The Function Point Counting Practices Manual" from IFPUG [9], was researched. The determination was made that there is not a need to re-iterate the rules specific to this research. Research indicates that care be taken when counting function points in an iterative project. One should be aware of deleting ILF's and adding EIF's, and double-counting, as has been shown to occur in iterative projects.

There are, however, two "local counting rules" that should be documented, as developed in cooperation with Tichenor [32]. These need to be documented because they pertain to an iterative development lifecycle, used in the FDD project. The first involves how to count "defect repair" conditions identified in an iteration and are repaired in future iterations.

1. *Defect Repair Rule*

When a defect (or number of defects) from an iteration is attempted to be repaired in a future iteration, the function count does not increase or decrease due to this activity. The repair activity does not represent new functionality, thus, the new enhancement should not consider the repair in the function point count for the new enhancement. The functionality was included in the function count of the iteration for which the defect was introduced.

The second "local counting rule" is applicable for cases when there is a single, countable low ILF (or EIF, EI, EO or EQ), and you choose to implement it partially in the first iteration and partially in the second, then it would count 3.5 function points for each iteration (or apportion it – i.e., if 1/3 of the DETs and RETs were done in iteration 1, count 1/3 of the function points for that iteration). This is because the ILF will not be completed until after the second iteration, but you want to account for the defects per function point and effort per function point after both the first and second iteration.

2. *Partial New Functionality per Iteration Rule*

When a function type is implemented over two iterations due to the choice of the software development style and not due to business reasons, its function point

count will be apportioned to each enhancement count per iteration. If the partial new functionality is wanted for business reasons, the developer should investigate the design to ensure that countable units are implemented in the same iteration so full functionality can be delivered to the user in all iterations. This is to ensure a purer representation/correlation of the defect per function ratio and effort per function point ratio per iteration. For example, if a new low ILF is developed in two equal parts over two iterations, count 3.5 unadjusted function points for each iteration. Ensure that the application function point count after the second iteration has been increased by 7 unadjusted function point counts, reflecting the complete new ILF, compared to the application function point count before the first of the two iterations started.

For the FDD project the lifecycle phases have been grouped to increase the readability of the function point graph (Table 19). These phases are grouped as:

Pre-Iteration
Group A = {RQ, RR, SD, SDR, SM, SMR, FL, FLR, FLP}

Iteration 0
Group B = {DE0, DR0, CO0, CI0, UT0, Igt0}
Group C = {ST0, AT0}

Iteration 1
Group D = {DE1, DR1, CO1, CI1, UT1, Igt1}
Group E = {ST1, AT1}

Iteration 2
Group F = {DE2, DR2, CO2, CI2, UT2, Igt2}
Group G = {ST2, AT2}

Iteration 3
Group H = {DE3, DR3, CO3, CI3, UT3, Igt3}
Group I = {ST3, AT3}

Table 19 FDD Project – Function Point Analysis

Business Function	Complexity (Factor)	Pre-It	It. 0		It. 1		It. 2		It. 3	
		Phases	Phases		Phases		Phase		Phases	
		A	B	C	D	E	F	G	H	I
ILF	Low (7)	70	37.45	37.45	2.25	53.2	2.8	56	14	70
	Avg. (10)	-	-	-	-	-	-	-	-	-
	High (15)	-	-	-	-	-	-	-	-	-
	Total	70	37.45	37.45	15.75	53.2	2.8	56	14	70
EIF	Low (5)	5	5	5	0	5	0	5	0	5
	Avg. (7)	-	-	-	-	-	-	-	-	-
	High (10)	-	-	-	-	-	-	-	-	-
	Total	5	5	5	0	5	0	5	0	5
EI	Low (3)	21	8.1	8.1	0.3	8.4	6.3	15	9	24
	Avg. (4)	8	2.4	2.4	0.8	3.2	4.8	7.6	4.8	8
	High (6)	6	3	3	1.8	4.8	-	5.4	-	6
	Total	35	13.5	13.5	2.9	16.4	11.1	28	13.8	38
EO	Low (4)	4	-	-	-	-	-	-	12	12
	Avg. (5)	5	-	-	-	-	5	5	-	5
	High (7)	-	-	-	-	-	-	-	-	-
	Total	9	0	0	0	0	5	5	12	17
EQ	Low (3)	12	3	3	-	3	3	6	9	12
	Avg. (4)	8	4	4	-	4	4	8	-	8
	High (6)	-	-	-	-	-	-	-	-	-
	Total	20	7	7	0	7	7	14	9	20
	UFP	139	62.95	62.95	18.65	81.6	25.9	108	48.8	150
	AFP	147.34	66.73	66.73	19.77	86.5	27.45	114.48	51.73	159.00

Table 20 General System Characteristics – FDD Project

General System Characteristic	Degree of Influence (0 to 5)
Data communications	0
Distributed data processing	2
Performance	5
Heavily used configuration	3
Transaction rate	5
Online data entry	5
End user efficiency	4
Online update	4
Complex processing	3
Reusability	0
Installation ease	2
Operational ease	3
Multiple sites	0
Facilitate change	5
Total	41

The values for total degree of influence, value adjustment factor and adjusted function point count are reflected in equations 3.1, 3.2 and 3.3. The AFP for the Pre-Iteration phases is shown as an illustration. The iterations, It. 0, It. 1, It. 2 and It. 3 use the same value of VAF to compute the AFP shown in Table 19. The VAF is based on the TDI. The project team and project characteristics remain the same throughout the project process, even though an iterative lifecycle is used. The features implemented for each iteration contain the same GSC values, thus, the TDI would be constant through each iteration causing VAF to remain constant.

TDI = 0 + 2 + 5 + 3 + 5 + 5 + 4 + 4 + 3 + 0 + 2 + 3 + 0 + 5 = 41 (3.1)

VAF = (TDI * 0.01) + 0.65 = (41 * 0.01) + 0.65 = 1.06 (3.2)

AFP = UFP * VAF = 139 * 1.06 = 147.34 (3.3)

In summary the function point count for the project and the iterations is as follows:

Function Point Count for the project start (Pre-Iteration) is:

147.34 (100 < 147.34 < 1000).

Function Point Count Iteration 0:	66.73
Function Point Count Iteration 0 Regression Testing (ST, AT):	66.73
Function Point Count Iteration 1:	19.77
Function Point Count Iteration 1 Regression Testing (ST, AT):	86.50
Function Point Count Iteration 2:	27.45
Function Point Count Iteration 2 Regression Testing (ST, AT):	114.48
Function Point Count Iteration 3:	51.73
Function Point Count Iteration 3 Regression Testing (ST, AT):	159.00

3.2.2 FDD Project Parameter Definitions and Derivations

The parameters used for the model in the FDD project are calculated from industry data, as in the Waterfall project. The Fault Introduction and Fault Removal phases use these values as input to the model. As in the Waterfall project, there is uncertainty in these industry parameters.

The parameters used in the model for the fault introduction phases are considered first. The following tables are required to calculate the value of $F_{j,\varphi}$.

Table 21 $\bar{t}_{fp,\varphi}$ Mean Effort per FP per Lifecycle Phase φ, in Staff Hours

Phase, φ	Max	Mode	Min	Mean *	Notes
RQ	2.64	0.75	0.38	1.00	Requirements
RR	1.76	0.59	0.33	0.74	Requirements Reviews
SD	2.64	0.75	0.33	1.00	Initial Design
SDR	1.76	0.59	0.33	0.74	Design Reviews
SM	1.32	0.44	0.26	0.56	Architecture
SMR	1.76	0.59	0.33	0.74	Design Reviews
FL	2.64	0.75	0.38	1.00	Requirements
FLR	1.76	0.59	0.33	0.74	Requirements Reviews
FLP	0.66	0.26	0.09	0.30	Project Plans
DE	5.28	0.88	0.44	1.54	Detailed Design
DR	1.76	0.59	0.33	0.74	Design Reviews
CO	8.80	2.64	0.66	3.34	Coding
CI	1.76	0.88	0.44	0.95	Code Inspection
UT	1.89	0.88	0.33	0.96	Unit Testing
IgT	1.76	0.75	0.33	0.85	Integration Testing
ST	1.32	0.66	0.26	0.70	System Testing
AT	1.76	0.38	0.22	0.58	Acceptance Testing

*** Note [12]:**
Mean from equation 2.38.

Next, the boundary conditions for (DP_{φ}), ($fd_{j,\varphi}$) and ($t_{FP,\varphi}$) are tabulated from the information above.

Table 22 Boundary Information for $DP_{\varphi} \bullet fd_{j,\varphi}$ and $\bar{t}_{FP,\varphi}$

	Requirements					
	RQ			FL		
	Max	Mean	Min	Max	Mean	Min
$DP_{\varphi} \bullet fd_{j,\varphi}$	1.50	1.00	0.40	1.50	1.00	0.40
$\bar{t}_{FP,\varphi}$	2.64	1.00	0.38	2.64	1.00	0.38
	Design					
	SD			SM		
	Max	Mean	Min	Max	Mean	Min
$DP_{\varphi} \bullet fd_{j,\varphi}$	2.20	1.25	0.60	2.20	1.25	0.60
$\bar{t}_{FP,\varphi}$	1.32	0.56	0.26	2.64	1.00	0.33
	Design					
	FLP			DE		
	Max	Mean	Min	Max	Mean	Min
$DP_{\varphi} \bullet fd_{j,\varphi}$	2.20	1.25	0.60	2.20	1.25	0.60
$\bar{t}_{FP,\varphi}$	0.66	0.30	0.09	5.28	1.54	0.44
	Coding					
	CO					
	Max	Mean	Min			
$DP_{\varphi} \bullet fd_{j,\varphi}$	2.50	1.75	1.00			
$\bar{t}_{FP,\varphi}$	8.80	3.34	0.66			

Based on equation 2.39 the values of the parameter, $F_{j,\varphi}$, for the Fault Introduction phases of RQ, FL, SD, SM, FLP, Dn, and COn, where n is the iteration number, are in Table 23.

Table 23 Values of $F_{j,\varphi}$ for the Fault Introduction Phases

	RQ	FL	SD	SM	FLP	DE	CO
$F_{j,\varphi}$	5.13	5.13	4.34	5.39	5.18	6.74	5.77

Using equation 2.40, the value for the fault potential per function point, Defect Potential (DP_{φ}) is determined. The Defect Potential for Pre-Iteration and Iterations 0, 1, 2 and 3 is determined. In the FDD project, the defect potential is from

Jones in 2008 [21]. The FDD project contains adjusted function point (AFP) values ranging from 19.77, for It. 1 to 147.34 for Pre-Iteration. We use the "Systems and Embedded Projects" category of [21] to get the range of defect potential values. The range for function point counts is:

10 < 19.77 < 27.45 < 51.73 < 66.73 < 100 < 147.34 < 1000

For Systems & Embedded Projects [21]:

Function Points = 10 Defect Potential = 2.50

Function Points = 100 Defect Potential = 3.00

Function Points = 1000 Defect Potential = 4.30

Next, the function point values for the iterations are interpolated to yield the DP.

Function Point Count Pre-Iteration: 147.34

$$DP(FDDFull) = 3.00 + \frac{4.30 - 3.00}{\log_{10}(1000) - \log_{10}(100)}\left(\log_{10}(147.34) - \log_{10}(100)\right) = 3.219$$

DP(FDDFull) = 3.219 (3.4)

Function Point Count Iteration 0: 66.73

$$DP(FDD) = 2.50 + \frac{3.00 - 2.50}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(66.73) - \log_{10}(10)\right) = 2.912$$

DP(FDDIt0) = 2.912 (3.5)

Function Point Count Iteration 1: 19.77

$$DP(FDD) = 2.50 + \frac{3.00 - 2.50}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(19.77) - \log_{10}(10)\right) = 2.648$$

DP(FDDIt1) = 2.648 (3.6)

Function Point Count Iteration 2: 27.45

$$DP(FDD) = 2.50 + \frac{3.00 - 2.50}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(27.45) - \log_{10}(10)\right) = 2.719$$

DP(FDDIt2) = 2.719 (3.7)

Function Point Count Iteration 3: 51.73

$$DP(FDD) = 2.50 + \frac{3.00 - 2.50}{\log_{10}(100) - \log_{10}(10)}\left(\log_{10}(51.73) - \log_{10}(10)\right) = 2.857$$

DP(FDDIt3) = 2.857 (3.8)

The defect potential does not need to have a weighted mean applied because the defect potential data is from 2008 and the FDD project was completed in 2011.

The value for the fraction of faults that originated in a phase φ, (fd_φ), is determined from Table 12 and categorized as a "Systems" project.

In summary, to calculate the parameters that constitute the Fault Introduction phases the following table is used.

Table 24 Summary of the Data Required to Calculate $\{v(t) * \mu_H(t)\}_{j,\varphi}$ – FDD Project

	Phase, φ									
	RQ	**RR**	**SD**	**SDR**	**SM**	**SMR**	**FL**	**FLR**	**FLP**	**DE0**
DP_φ	3.219		3.219		3.219		3.219		3.219	2.912
$fd_{j,\varphi}$	0.10	n/a	0.25	n/a	0.25	n/a	0.10	n/a	0.25	0.25
$t_{FP,\varphi}$, in staff hours	1.00	0.74	1.00	0.74	0.56	0.74	1.00	0.74	0.30	1.54
	Phase, φ									
	DR0	**CO0**	**CI0**	**UT0**	**IgT0**	**AT0**	**ST0**	**DE1**	**DR1**	**CO1**
DP_φ		2.912						2.648		2.648
$fd_{j,\varphi}$	n/a	0.40	n/a	n/a	n/a	n/a	n/a	0.25	n/a	0.40
$T_{FP,\varphi}$, in staff hours	0.74	3.34	0.95	0.96	0.85	0.58	0.70	1.54	0.74	3.34
	Phase, φ									
	CI1	**UT1**	**IgT1**	**AT1**	**ST1**	**DE2**	**DR2**	**CO2**	**CI2**	**UT2**
DP_φ						2.719		2.719		
$fd_{j,\varphi}$	n/a	n/a	n/a	n/a	n/a	0.25	n/a	0.40	n/a	n/a
$T_{FP,\varphi}$, in staff hours	0.95	0.96	0.85	0.58	0.70	1.54	0.74	3.34	0.95	0.96
	Phase, φ									
	IgT2	**AT2**	**ST2**	**DE3**	**DR3**	**CO3**	**CI3**	**UT3**	**IgT3**	**AT3**
DP_φ				2.857		2.857				
$fd_{j,\varphi}$	n/a	n/a	n/a	0.25	n/a	0.40	n/a	n/a	n/a	n/a
$t_{FP,\varphi}$, in staff hours	0.85	0.58	0.70	1.54	0.74	3.34	0.95	0.96	0.85	0.58
	Phase, φ									
	ST3									
DP_φ										
$fd_{j,\varphi}$	n/a									
$T_{FP,\varphi}$, in staff hours	0.70									

The SLI value used with the $F_{j,\varphi}$ parameter is computed as discussed in previous sections. An expert opinion elicitation was performed to assess the quality of the influencing factors of Table 2 for the FDD project. The results of the elicitation are given in Appendix C.

The fault removal component for Fault Introduction phases, in FDD, follows equation 2.22 for the removal rate. The fault removal component for the Fault Removal phases, in FDD, follows equation 2.32 for the removal rate.

3.2.3 FDD Project Data

The data collected for the FDD project consists of actual staff hours and observed faults in selected phases, as in the Waterfall project. The SLI, when used, is again calculated from expert opinion.

The mean effort in staff hours per function point per phase is displayed in the following table (Table 25), which is data gathered from industry. Following the table is the estimated staff hours per phase as calculated from the function point value and mean effort value.

Table 25 Mean Effort In Staff Hours Per Function Point Per Phase ($t_{FP,\varphi}$) – FDD Project

Activity	**RQ**	**RR**	**SD**	**SDR**	**SM**	**SMR**	**FL**
Effort	1.00	0.74	1.00	0.74	0.56	0.74	1.00
Activity	**FLR**	**FLP**	**DE**	**DR**	**CO**	**CI**	**UT**
Effort	0.74	0.30	1.54	0.74	3.34	0.95	0.96
Activity	**IgT**	**ST**	**AT**				
Effort	0.85	0.70	0.58				

Table 26 Lifecycle Phase In Estimated Staff Hours from Function Points – FDD Project

Activity	Effort	Function Points	Staff Hours
RQ	1.00	147.34	147.34
RR	0.74	147.34	109.03
SD	1.00	147.34	147.34
SDR	0.74	147.34	109.03
SM	0.56	147.34	82.51
SMR	0.74	147.34	109.03
FL	1.00	147.34	147.34
FLR	0.74	147.34	109.03
FLP	0.30	147.34	44.2
DE0	1.54	66.73	102.76
DR0	0.74	66.73	49.38
CO0	3.34	66.73	222.88
CI0	0.95	66.73	63.39
UT0	0.96	66.73	64.06
IgT0	0.85	66.73	56.72
ST0	0.70	66.73	46.71
AT0	0.58	66.73	38.7
DE1	1.54	19.77	30.45
DR1	0.74	19.77	14.63
CO1	3.34	19.77	66.03
CI1	0.95	19.77	18.78
UT1	0.96	19.77	18.98
IgT1	0.85	19.77	16.8
ST1	0.70	86.50	60.55
AT1	0.58	86.50	50.17
DE2	1.54	27.45	42.27
DR2	0.74	27.45	20.31
CO2	3.34	27.45	91.68
CI2	0.95	27.45	26.08
UT2	0.96	27.45	26.35
IgT2	0.85	27.45	23.33
ST2	0.70	114.48	80.14
AT2	0.58	114.48	66.4
DE3	1.54	51.73	79.66
DR3	0.74	51.73	29.65
CO3	3.34	51.73	172.78
CI3	0.95	159.00	151.05
UT3	0.96	51.73	49.66
IgT3	0.85	51.73	43.97
ST3	0.70	159.00	111.3
AT3	0.58	159.00	92.22

Table 27 Lifecycle Phase In Staff Hours (Estimated and Actual) – FDD Project

Activity	Staff Hours (From Function Points)	Staff Hours (Actual)	Cumulative Staff Hours (Actual)
RQ	147.34	171.0	171.00
RR	109.03	65.5	236.50
SD	147.34	99.5	336.00
SDR	109.03	32.0	368.00
SM	82.51	8.0	376.00
SMR	109.03	1.94	377.94
FL	147.34	121.0	498.94
FLR	109.03	12.0	510.94
FLP	44.2	12.0	522.94
DE0	102.76	122.5	645.44
DR0	49.38	23.0	668.44
CO0	222.88	549.3	1,217.74
CI0	63.39	50.0	1,267.74
UT0	64.06	183.1	1,450.84
IgT0	56.72	183.1	1,633.94
ST0	46.71	127.0	1,760.94
AT0	38.70	60.0	1,820.94
DE1	30.45	104.0	1,924.94
DR1	14.63	16.0	1,940.94
CO1	66.03	108.0	2,048.94
CI1	18.78	24.0	2,072.94
UT1	18.98	36.0	2,108.94
IgT1	16.8	36.0	2,144.94
ST1	60.55	82.0	2,226.94
AT1	50.17	144.5	2,371.44
DE2	42.27	108.0	2,479.44
DR2	20.31	29.0	2,508.44
CO2	91.68	165.0	2,673.44
CI2	26.08	30.0	2,703.44
UT2	26.35	55.0	2,758.44
IgT2	23.33	55.0	2,813.44
ST2	80.14	88.0	2,901.44
AT2	66.4	80.0	2,981.44
DE3	79.66	265.5	3,246.94
DR3	29.65	54.0	3,300.94
CO3	172.78	705.0	4,005.94
CI3	151.05	286.25	4,292.19
UT3	49.66	235.0	4,527.19
IgT3	43.97	235.0	4,762.19
ST3	111.30	116.0	4,878.19
AT3	92.22	160.0	5,038.19
Total	3,032.69	5,038.19	

A comparison to the actual data reveals variations per phase between the actual and estimated hours.

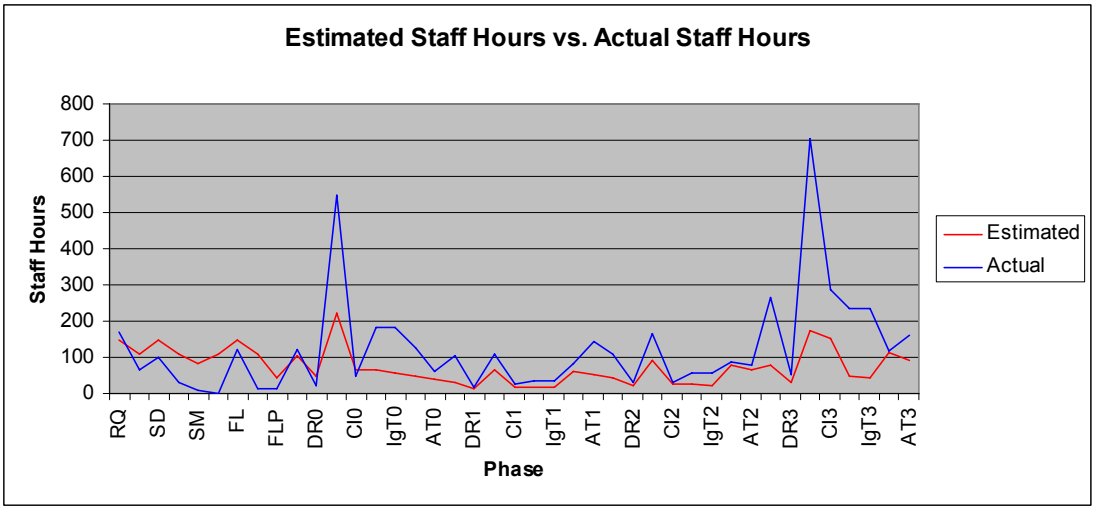


Figure 19 Estimated vs. Actual Staff Hours – FDD Project

A conclusion to be drawn from the data in Table 27 and the figure above (Figure 19) reveals that overall in most phases the estimate from industry data was too small. This was particularly evident in coding and testing phases. Coding phases were most under-estimated, based on actual hours. The trend in the actual data appears to be that less effort is spent in requirements and design phases and more effort in coding and testing phases.

Faults were observed and recorded in thirteen phases, Requirements Review (RR), System Design Review (SDR), System Model Review (SMR), Feature List Review (FLR), Design Review for iterations 0 through 3 (DRn), n is the iteration number, System Test for iterations 0 through 3 (STn) and Code Inspection for iteration 3 (CI3), which was a complete system code inspection. For the RR and FLR phases, the faults recorded were requirements faults. For the SDR, SMR and DRn phases, the faults recorded were design faults, as they were the category of faults

reviewed during the design review process of this project. In the STn phase, all categories of faults; requirements, design and coding faults, were recorded as a composite value. In the CI3 phase, the faults recorded were coding faults.

Table 28 Observed & Repaired Faults Found per Phase (FDD)

Activity	Number of Faults Actually Observed & Repaired		
	Category 1. Critical	**Category 2. Major**	**Total**
RQ Faults - Requirements Review (RR)	1	1	2
DE Faults – System Design Review (SDR)	0	1	1
DE Faults – System Model Review (SMR)	0	0	0
RQ Faults – Feature List Review (FLR)	0	1	1
DE Faults – Design Review Iteration 0 (DR)	3	1	4
RQ, DE, CO Faults - System Test Iteration 0 (ST)	8	0	8
DE Faults – Design Review Iteration 1 (DR)	3	0	3
RQ, DE, CO Faults - System Test Iteration 1 (ST)	4	0	4
DE Faults – Design Review Iteration 2 (DR)	0	1	1
RQ, DE, CO Faults - System Test Iteration 2 (ST)	1	0	1
DE Faults – Design Review Iteration 3 (DR)	2	6	8
CO Faults – Code Inspection Iteration 3 (CI)	5	16	21
RQ, DE, CO Faults - System Test Iteration 3 (ST)	0	1	1
Total	**27**	**28**	**55**

3.2.4 FDD Project Software Reliability Model Results

The reliability model was applied, with the preceding data, to the FDD project. The fault predictions of the reliability model are based on the industry-based metrics, the calculated metrics based on the industry-based metrics and expert opinion-based parameters. At this point, the expert-opinion parameter, SLI, is only applied to the Fault Introduction phases. Refer to Appendix C for the calculation of the SLI from expert opinion for the FDD project. The SLI used is based on expert opinion; an enhancement to the model will make the SLI parameter available to Fault Removal phases in Chapter 4. Another extension to the model will optimize the SLI parameter based on the observable data also in Chapter 4. The staff hours parameter used in the model is the actual staff hours and not the estimated staff hours based on industry data. The parameters for this test case (#27) are given in Appendix D.

The output from the model includes the value of the requirements, design, coding and cumulative faults at each phase in the development cycle.

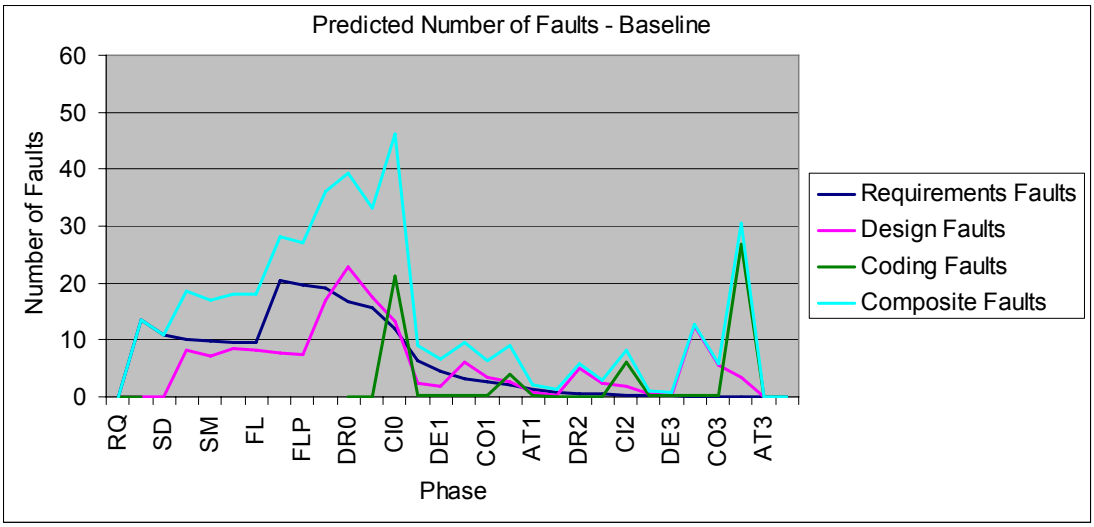


Figure 20 Model Predicted Fault Count – FDD Lifecycle

Table 29 Model Expected Observed Faults Vs. Observed Faults – FDD Lifecycle

Inspection Phase	Fault Type	Expected Faults	Observed & Repaired Faults
RR	RQ Faults	2.62	2
SDR	DE Faults	1.26	1
SMR	DE Faults	0.08	0
FLR	RQ Faults	0.79	1
DR0	DE Faults	5.21	4
ST0	Cumulative Faults	2.75	8
DR1	DE Faults	2.80	3
ST1	Cumulative Faults	0.36	4
DR2	DE Faults	2.78	1
ST2	Cumulative Faults	0.18	1
DR3	DE Faults	6.78	8
CI3	CO Faults	26.46	21
ST3	Cumulative Faults	0.01	1
Total		52.08	55
Percent Error (Expected vs. Observed Faults)		5.31	

The reliability model, with the parameters as above, predicts that 0.063 faults could be expected in the system at the end of development. This number will change as enhancements and optimizations are applied to the model in Chapter 4.

The “Percent Error” measure is defined as the per cent of absolute value of: (Expected Observed Faults – Actual Observed Faults) / Actual Observed Faults This measure is applied to the total expected observed and actual observed faults. The percent error in the measurement illustrates the measure of distance of the total expected observed faults from the model predictions to the actual observed faults. The percent error is valuable in determining performance of the model as a means of predicting observed faults.

Table 29 indicates that the model parameter values predicted lower fault values than observed which indicates improvements may be made. These enhancements and optimizations will be performed in Chapter 4.

Chapter 4: Enhancements and Optimizations

The first enhancement to be introduced to the model is the insertion of the SLI parameter into the Fault Removal phase.

The second enhancement uses an analysis of the model to determine an optimal value for SLI. The optimization can occur for a fault introduction phase and its immediately subsequent fault removal phase based on the actual observed faults of the removal phase. For the optimization, actual observed fault data is necessary.

4.1 Insertion of SLI into Fault Removal Phases

The SLI parameter can be inserted into the Fault Removal phases of the model. To obtain this insertion, a value of Z_a which is calculated from the max, min and mean of Z_a is found.

The value of Z_a for a Fault Removal phase is given in equation 2.32.

$$\{z_a(t)\}_{j,\varphi} = -\ln\left[1 - DRE_{\varphi}\right] \bullet \frac{1}{t_{FP,\varphi} \bullet N_{FP}} \tag{4.1}$$

The max and min values are given by:

$$\{z_a(t)\}_{j,\varphi,Max} = -\ln\left[1 - DRE_{\varphi,Max}\right] \bullet \frac{1}{t_{FP,\varphi} \bullet N_{FP}} \tag{4.2}$$

$$\{z_a(t)\}_{j,\varphi,Min} = -\ln\left[1 - DRE_{\varphi,Min}\right] \bullet \frac{1}{t_{FP,\varphi} \bullet N_{FP}} \tag{4.3}$$

Where:

$\{z_a(t)\}_{j,\varphi}$	intensity function of per-fault detection in phase φ
$t_{FP,\varphi}$	effort necessary to review a function point in phase φ
DRE_{φ}	Defect Removal Efficiency in phase φ

N_{FP} Number of function points

The effort necessary to review a function point, $t_{FP,\varphi}$ is used. This is consistent with the effort component in Fault Introduction phases. In Fault Introduction phases the $F_{j,\varphi}$ component of the phase is calculated using the minimum and maximum values of DP_{φ}, $fd_{j,\varphi}$ and $t_{FP,\varphi}$. In the Fault Introduction phase equation (2.37) the mean value of $t_{FP,\varphi}$ is used. This is consistent with the use of $t_{FP,\varphi}$ in equations 4.2 and 4.3, which are components of the Fault Removal phase in equation 4.13. The removal efficiency, DRE_{φ} is varying between a minimum and maximum efficiency, but, the effort, $t_{FP,\varphi}$, is consistent. The enhancement, SLI in removal phases, is showing the effect of the varying human component, the SLI, on the defect removal efficiency, DRE_{φ}.

There are some obstacles to using SLI in the Fault Removal phases. First, the ranges for DRE given in industry data do not cover all of the phases in a lifecycle. Second, the lifecycle phases which have ranges for DRE are not given as removal efficiency per defect origin, but rather as, removal efficiency per phase.

The following relationship exists based on the assumption that defect removal is a human endeavor and is governed by human error. As shown in equation 4.1, Z_a is related to the DRE value and the length of the phase, T. T is the mean effort per function point $t_{FP,\varphi}$ multiplied by the number of function points in the phase N_{FP}. Thus, the human error probability, HEP, is related to - Z_aT through equation 4.1 and the DRE human error component. One of the key assumptions of SLI, as given in Chapter 2, is that the probability of a human error is logarithmically proportional to the SLI. This assumption indicates the relationship with SLI in equation 4.4.

$$-\{z_a\}_{j,\varphi} T_\varphi = \log HEP = aSLI_\varphi + b \tag{4.4}$$

If SLI is set to its minimum and maximum values we get the following equations.

SLI = 1; $$-\{z_{aMAX}\}_{j,\varphi} T_\varphi = a \bullet 1 + b \tag{4.5}$$

SLI = 0; $$-\{z_{aMIN}\}_{j,\varphi} T_\varphi = b \tag{4.6}$$

Using these relationships:

$$a = -\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] T_\varphi \tag{4.7}$$

$$-\{z_a\}_{j,\varphi} T_\varphi = -\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] T_\varphi \bullet SLI_\varphi - \{z_{aMIN}\}_{j,\varphi} T_\varphi \tag{4.8}$$

$$\{z_a\}_{j,\varphi}(SLI) = \left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] \bullet SLI_\varphi + \{z_{aMIN}\}_{j,\varphi} \tag{4.9}$$

Defining $\overline{z_a}$ as the value of z_a for an average development environment (i.e., SLI_φ = 0.5) we solve for z_a as a function of $\overline{z_a}$:

$$\{\overline{z_a}\}_{j,\varphi} = \{z_{aMIN}\}_{j,\varphi} + 0.5\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] \tag{4.10}$$

$$\{z_a\}_{j,\varphi} = \{z_{aMIN}\}_{j,\varphi} + SLI_\varphi\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] \tag{4.11}$$

$$\{z_a\}_{j,\varphi} - \{\overline{z_a}\}_{j,\varphi} = \left(SLI_\varphi - 0.5\right)\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] \tag{4.12}$$

$$\{z_a\}_{j,\varphi} = \{\overline{z_a}\}_{j,\varphi} + \left(SLI_\varphi - 0.5\right)\left[\{z_{aMAX}\}_{j,\varphi} - \{z_{aMIN}\}_{j,\varphi}\right] \tag{4.13}$$

Where:

$\{\overline{z_a}\}_{j,\varphi}$	the average Z_a for phase φ
$\{z_{aMAX}\}_{j,\varphi}$	the highest Z_a , DRE for phase φ
$\{z_{aMIN}\}_{j,\varphi}$	the lowest Z_a , DRE for phase φ
SLI_φ	The SLI for phase φ

The DRE ranges are obtained from Jones [21] in the following table.

Table 30 Ranges Of Defect Removal Efficiency

Removal Phase	Removal Step	Efficiency (%)		
		Lowest	Modal	Highest
RR, FLR	Requirements review	**20**	**30**	**50**
SDR	Top-level design reviews	**30**	**40**	**60**
SMR, DR	Detailed functional design inspections	**30**	**45**	**65**
	Detailed logic design inspections	**35**	**55**	**75**
CI	Code inspections	**35**	**60**	**85**
	Modeling or prototyping	**35**	**65**	**80**
	Unit tests	**10**	**25**	**50**
	New function tests	**20**	**35**	**55**
	Integration tests	**25**	**45**	**60**
ST	System test	**25**	**50**	**65**
	Cumulative efficiency	**75**	**97**	**99.99**

4.1.1 Waterfall Project Software Reliability Model Results with Fault Removal SLI

The reliability model, with the enhancement of the SLI in the Fault Removal phases, is applied to the Waterfall project. In this model, the SLI is a component of the Fault Removal phases, RR, DR and ST. The SLI for the phases RQ, DE, CO, RR, DR and ST are based on expert opinion from the tables given in Appendix B. The staff hours parameters used in the model are from actual staff hours and not the estimated staff hours. The model is the same as in Chapter 2, except for the additional SLI components for the RR, DR and ST Fault Removal phases. The parameters for this test case (#41) are given in Appendix D.

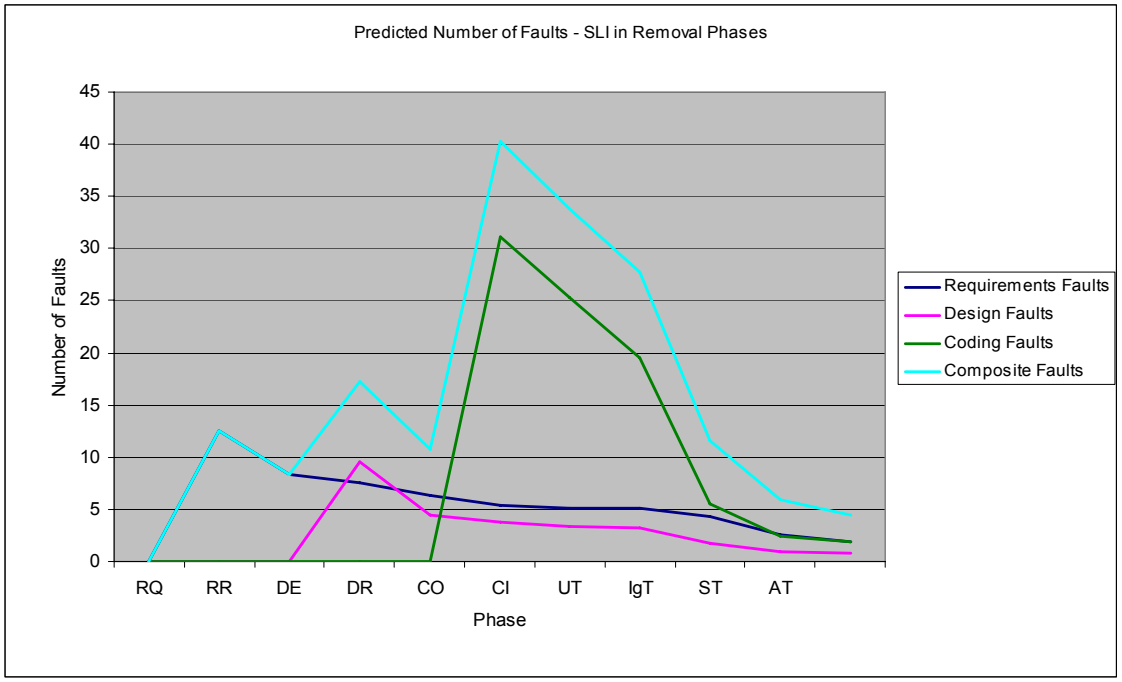


Figure 21 Model Predicted Fault Count – Waterfall Project – SLI in Removal Phases

Table 31 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle – SLI in Removal Phases

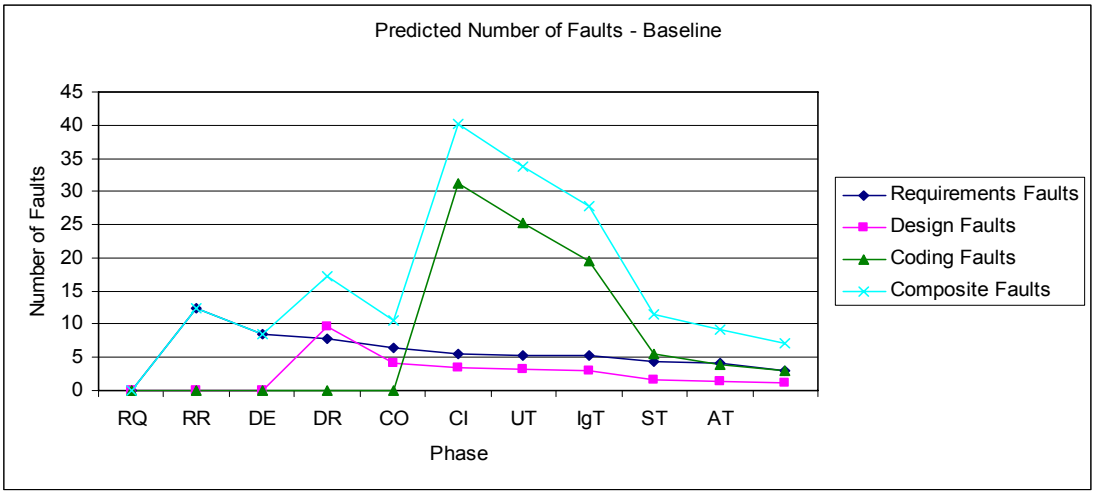
Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phase Expected Faults	Observed & Repaired Faults
RR	RQ Faults	4.05	4.20	6
DR	DE Faults	5.54	5.22	7
ST	Cumulative Faults	2.26	5.69	7
Total		11.85	15.11	20
Percent Error (Expected vs. Observed Faults)		40.75	24.45	

The reliability model, with the parameters as above, predicts that 4.48 faults could be expected in the system at the end of development. The overall predicted

faults left in the system dropped from 7.06 to 4.48. This number is a decrease in the predicted number of observed faults left in the system at the end of development. Also, the expected faults found have less per cent error to the observed & repaired faults, as shown in Table 31 than the previous model. As expected the Fault Introduction values remained unchanged. Of the Fault Removal phases the RR and ST increased their fault detection rate and the DR phase had a slightly lower fault detection rate. There were 4.38 design faults remaining in the system after DR with SLI as a fault removal component, as opposed to 4.06. The percent error between the expected observed and actually observed faults dropped from 41% to 24%. The SLI values for RR, DR and ST come from the data given in Appendix B, as do the SLI values for the Fault Introduction phases. All the SLI values are from the elicitation of expert opinion from the project team.

The output of the model, as shown in Table 31, indicates improvements could be made in the model parameters. Table 31 indicates that the model parameter values predicted lower fault values than observed, although better values than without including the fault removal SLI component. Optimizing the SLI values to the observed fault data rather than solely on expert opinion is the next step in improving the expert opinion assessment, thus this is the next enhancement.

A side by side comparison of the chart of predicted fault count from the Baseline model and the predicted fault count from the SLI in Removal Phase model follows in Figure 22. The comparison shows slight variation after the RR, DR and ST phases as a result of using the SLI based on the elicitation of experts from the project team for these phases.



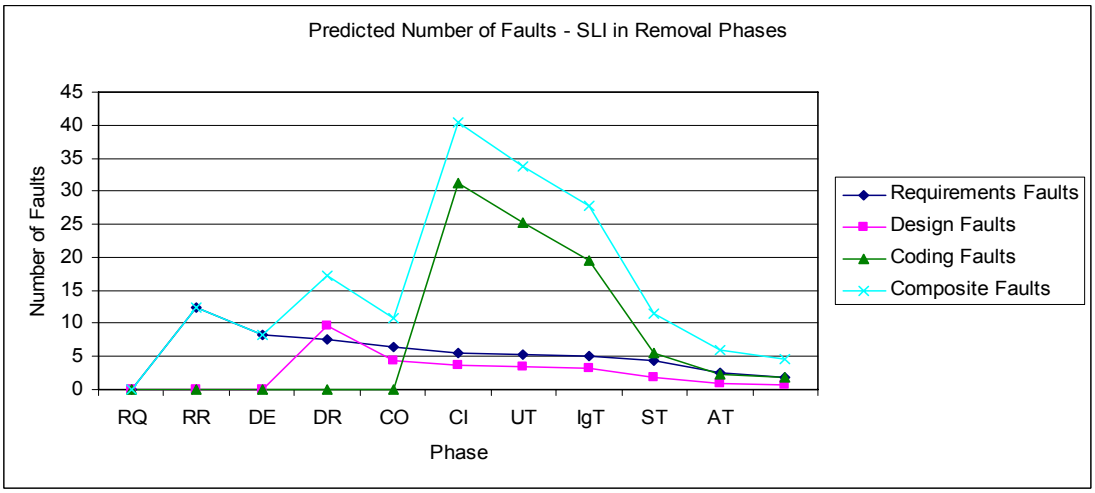


Figure 22 Comparison of Predicted Number of Faults – Baseline and SLI in Removal Phases Models– Waterfall

Table 32 shows the use of SLI in the removal phases. The removal phases in which an SLI was obtained from expert opinion were used in this model. The CI, UT, IgT and AT phases did not have fault data, in this project, thus, an SLI from expert opinion value was not obtained for those phases.

Table 32 SLI Values – Baseline and SLI in Removal Phases Models - Waterfall

Activity	SLI - Baseline	SLI – In Removal Phases	SLI from Expertise
RQ	0.213	0.213	0.213
RR	No SLI	0.6623	0.6623
DE	0.413	0.413	0.413
DR	No SLI	0.5361	0.5361
CO	0.383	0.383	0.383
CI	No SLI	No SLI	Not Done
UT	No SLI	No SLI	Not Done
IgT	No SLI	No SLI	Not Done
ST	No SLI	0.675	0.675
AT	No SLI	No SLI	Not Done

4.1.2 *FDD Project Software Reliability Model Results with Fault Removal SLI*

The reliability model is applied to the FDD project. In this model, the SLI is a component of the Fault Removal phases, RR, SDR, SMR, FLR, DRn (n = iteration number) and STn. The SLI for the Fault Removal phases as well as the Fault Introduction phases RQ, SD, SM, FL, DEn and COn are based on expert opinion from the tables given in Appendix B. The staff hours parameters used in the model are from actual staff hours and not the estimated staff hours from industry data.

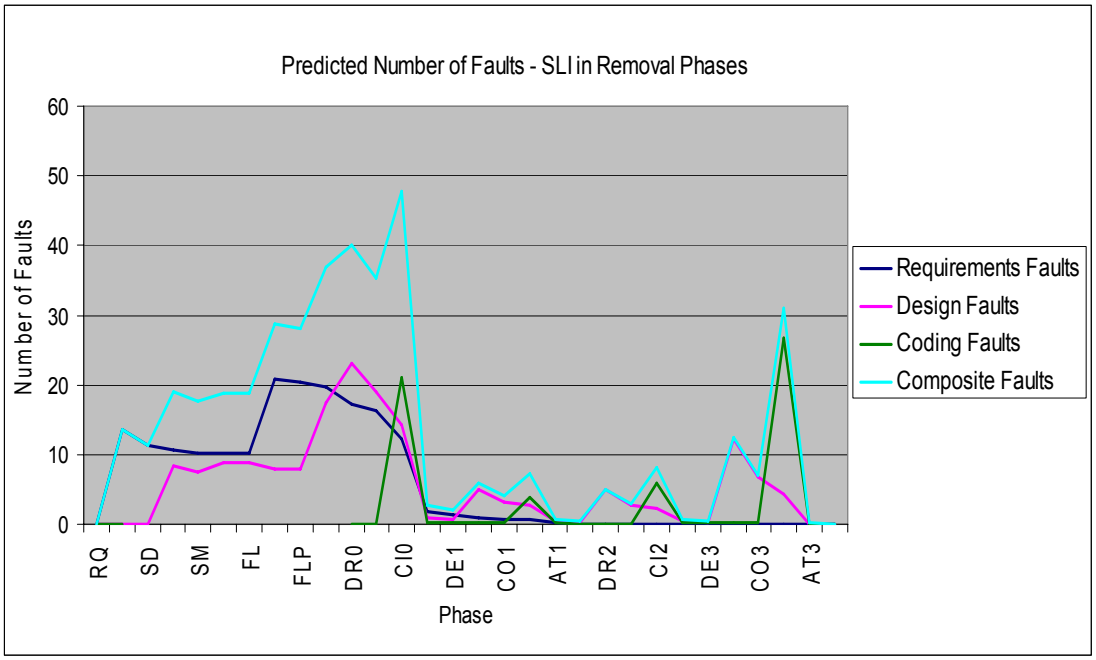


Figure 23 Model Predicted Fault Count – FDD Project – SLI in Removal Phases

Table 33 Model Expected Observed Faults Vs. Observed Faults – FDD Lifecycle – SLI in Removal Phases

Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phases Expected Faults	Observed & Repaired Faults
RR	RQ Faults	2.62	2.12	2
SDR	DE Faults	1.26	0.93	1
SMR	DE Faults	0.08	0.06	0
FLR	RQ Faults	0.79	0.62	1
DR0	DE Faults	5.21	4.04	4
ST0	Cumulative Faults	2.75	9.60	8
DR1	DE Faults	2.80	1.80	3
ST1	Cumulative Faults	0.36	0.78	4
DR2	DE Faults	2.78	2.17	1
ST2	Cumulative Faults	0.18	0.48	1
DR3	DE Faults	6.78	5.46	8
CI3	CO Faults	26.46	19.81	21
ST3	Cumulative Faults	0.01	0.11	1
Total		52.08	47.98	55
Percent Error (Expected vs. Observed Faults)		5.31	12.76	

The parameters for this test case (#28) are given in Appendix D. The reliability model, with the parameters as above, predicts that 0.095 faults could be expected in the system at the end of development. This number compares similarly to the model application without the SLI component in the Fault Removal phases with 0.063 faults remaining. The SLI for RR, SDR, SMR, FLR, DR0, ST0, DR1, ST1, DR2, ST2, DR3, CI3 and ST3 come from the data given in Appendix C. The percent error between the expected observed and actually observed faults rose from 5% to

13%. The Fault Introduction values remained unchanged. Of the Fault Removal phases the ST phases showed increased fault detection rate and the other phases had a slightly lower fault detection rate resulting in the lower total model expected observed faults.

The output of the model, as shown in Table 33, indicates improvements could be made in the model parameters. Table 33 indicates that the model parameters predicted lower total fault values than observed. As in the Waterfall project, optimizing the SLI values to the observed fault data rather than solely on expert opinion is the next step in improving the expert opinion assessment, thus this is the next enhancement.

A side by side comparison of the chart of predicted fault count from the Baseline model and the predicted fault count from the SLI in Removal Phase model follows in Figure 24. The comparison shows little variation in all phases as a result of using the SLI based on the elicitation of experts from the project team for the removal phases.

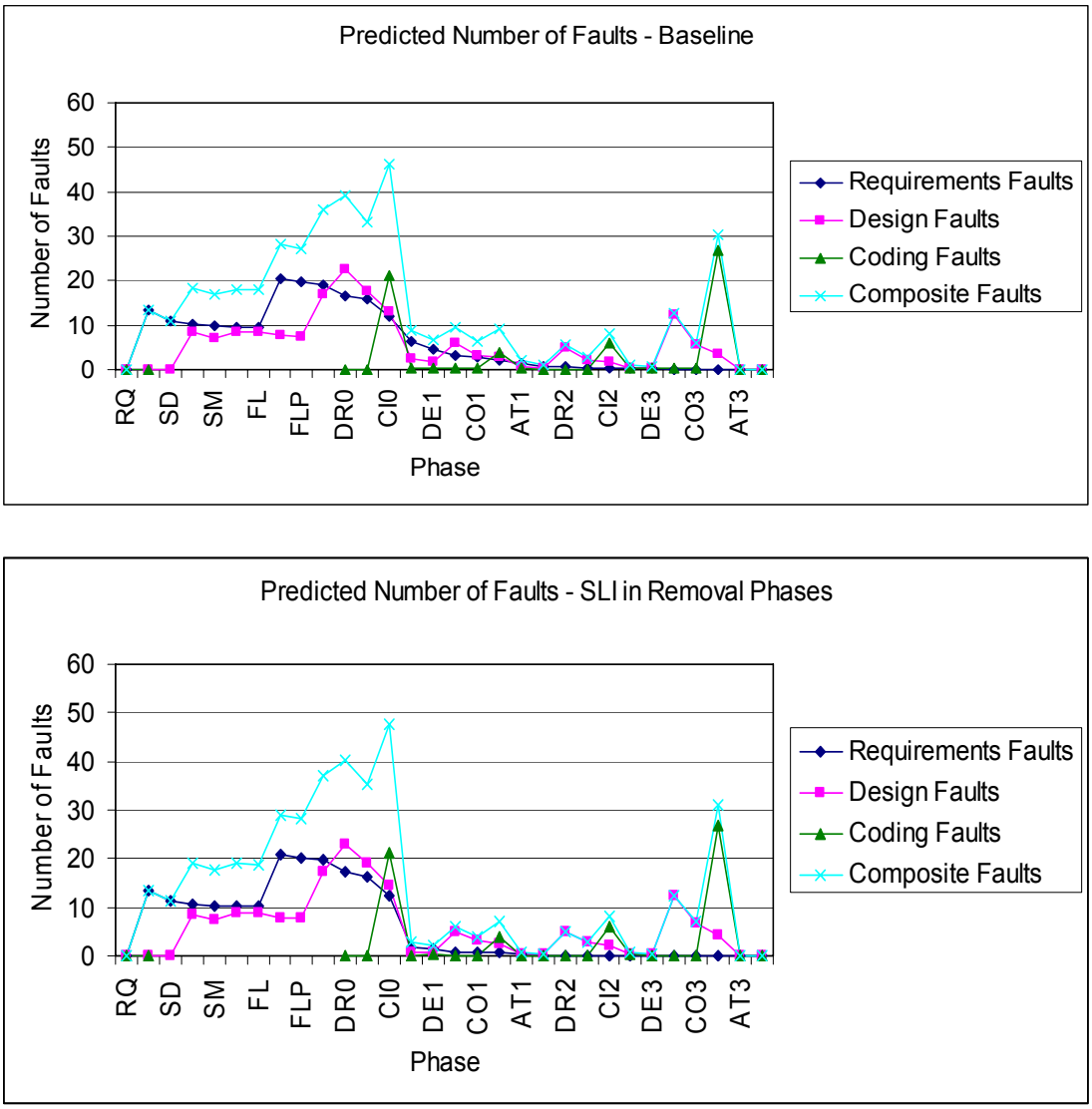


Figure 24 Comparison of Predicted Number of Faults – Baseline and SLI in Removal Phases Models– FDD

Table 34 shows the use of SLI in the removal phases. The removal phases in which an SLI was obtained from expert opinion were used in this model. The UT, IgT and AT phases did not have fault data, in this project, thus, an SLI from expert opinion value was not obtained for those phases.

Table 34 SLI Values – Baseline and SLI in Removal Phases Models - FDD

Activity	SLI - Baseline	SLI – In Removal Phase	SLI from Expertise
RQ	0.4259	0.4259	0.4259
RR	No SLI	0.3852	0.3852
SD	0.7093	0.7093	0.7093
SDR	No SLI	0.3889	0.3889
SM	0.6463	0.6463	0.6463
SMR	No SLI	0.3093	0.3093
FL	0.3889	0.3889	0.3889
FLR	No SLI	0.363	0.363
FLP	0.3704	0.3704	0.3704
DE0	0.6778	0.6778	0.6778
DR0	No SLI	0.3389	0.3389
CO0	0.6593	0.6593	0.6593
CI0	No SLI	No SLI	0.3981
UT0	No SLI	No SLI	Not Done
IgT0	No SLI	No SLI	Not Done
ST0	No SLI	0.6556	0.6556
AT0	No SLI	No SLI	Not Done
DE1	0.6778	0.6778	0.6778
DR1	No SLI	0.3389	0.3389
CO1	0.6593	0.6593	0.6593
CI1	No SLI	No SLI	0.3981
UT1	No SLI	No SLI	Not Done
IgT1	No SLI	No SLI	Not Done
ST1	No SLI	0.6556	0.6556
AT1	No SLI	No SLI	Not Done
DE2	0.6778	0.6778	0.6778
DR2	No SLI	0.3389	0.3389
CO2	0.6593	0.6593	0.6593
CI2	No SLI	No SLI	0.3981
UT2	No SLI	No SLI	Not Done
IgT2	No SLI	No SLI	Not Done
ST2	No SLI	0.6556	0.6556
AT2	No SLI	No SLI	Not Done
DE3	0.6778	0.6778	0.6778
DR3	No SLI	0.3389	0.3389
CO3	0.6593	0.6593	0.6593
CI3	No SLI	0.3981	0.3981
UT3	No SLI	No SLI	Not Done
IgT3	No SLI	No SLI	Not Done
ST3	No SLI	0.6556	0.6556
AT3	No SLI	No SLI	Not Done

4.2 Optimization of the SLI Parameter

Previous sections discuss the SLI parameter and its evaluation based on expert opinion. These sections also introduce the concept of optimizing the value of this parameter. By introducing the SLI parameter into the Fault Removal phases, optimization of SLI in all Fault Removal phases was made possible. To optimize the value of SLI, analysis of the model equation was performed. The goal of the analysis was to determine, through optimizations based on actual observed data, the SLI values in the Fault Introduction and Fault Removal phases.

The result of optimizing the SLI values is to more accurately assess their actual values using information available directly from the life cycle. Another insight gained from the optimization is the resulting control of the model through the SLI parameter.

To optimize the SLI parameter an analysis was performed to derive equations in which dependence in the SLI is made explicit.

4.2.1 Analysis of the Model Equation Solving for the Value of Actual Faults (μ_U Observed)

Equation 2.6 expresses the expected number of faults present in the system at a time, t. The predicted number of faults of type "j" detected or observed during phase φ is expressed as:

$$\left\{\mu_U(t)\right\}_{j,\varphi}^{Observed\ \Pr edicted} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \left\{z_a\right\}_{j,\varphi} \left\{\mu_U(s)^{\Pr edicted}\right\}_{j,\varphi} ds \qquad (4.14)$$

In order to obtain this value the predicted value of μ_U is derived first. The value of Z_a is known in Fault Introduction (equation 2.22) and Fault Removal (equation 2.32) phases. Equation 2.6 is solved for the Fault Introduction phases.

The predicted value of μ_U, for Fault Introduction phases, is derived from equation 2.6, after manipulation the equation form is thus:

$$\frac{d\mu_U(t)_{j,\varphi}}{dt} - \{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}\{\mu_U(t)\}_{j,\varphi} = \{\nu(t)\mu_H(t)\}_{j,\varphi} \quad (4.15)$$

Using the solution method:

$$\frac{d\mu_U(t)_{j,\varphi}}{dt} + P(t)\mu_U(t)_{j,\varphi} = Q(t) \quad (4.16)$$

$$e^{\int P(t)dt}\mu_U(t)_{j,\varphi} = \int e^{\int P(s)ds}Q(t)dt + C \quad (4.17)$$

Where:

j	a category of faults introduced during phase j, ie., j = RQ, DE or CO
φ	a lifecycle phase, ie., φ = RQ, RR, DE, DR, CO, CI, UT, IgT, ST, AT
t	software development lifecycle time

$$P(t) = -\left(\{z_a(t)\}_{j,\varphi} \bullet \{\mu_R(t)\}_{j,\varphi}\right) \quad (4.18)$$

$$Q(t) = \{\nu(t)\mu_H(t)\}_{j,\varphi} \quad (4.19)$$

$$\{z_a(t)\}_{j,\varphi} = \frac{\frac{DRE_\varphi}{1 - DRE_\varphi}}{\left[t - t_{Begin\varphi}\right]} \quad (4.20)$$

$$\int P(t)dt = -\int \frac{\left(\frac{DRE_{\varphi}}{1-DRE_{\varphi}}\right) \bullet \{\mu_R\}_{j,\varphi}}{\left[t - t_{Begin\varphi}\right]} dt \tag{4.21}$$

$$= -\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{1-DRE_{\varphi}}\right) \int \frac{1}{\left[t - t_{Begin\varphi}\right]} dt$$

$$= \left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) \ln\left(\left[t - t_{Begin\varphi}\right]\right)$$

From the solution method applied above, equation 4.17 becomes:

$$\{\mu_U(t)\}_{j,\varphi} \bullet e^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) \ln\left(\left[t - t_{Begin\varphi}\right]\right)} = \int \{\nu \bullet \mu_H\}_{j,\varphi} \bullet e^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) \ln\left(\left[t - t_{Begin\varphi}\right]\right)} dt \tag{4.22}$$

Since $\{\nu \bullet \mu_H\}_{j,\varphi}$ and $\{\mu_R\}_{j,\varphi}$ are constant over phase φ:

$$\{\mu_U(t)\}_{j,\varphi} \bullet e^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) \ln\left(\left[t - t_{Begin\varphi}\right]\right)} = \{\nu \bullet \mu_H\}_{j,\varphi} \int e^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) \ln\left(\left[t - t_{Begin\varphi}\right]\right)} dt \tag{4.23}$$

$$\{\mu_U(t)\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right)} = \{\nu \bullet \mu_H\}_{j,\varphi} \int \left[t - t_{Begin\varphi}\right]^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right)} dt \tag{4.24}$$

$$\{\mu_U(t)\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right)} = \{\nu \bullet \mu_H\}_{j,\varphi} \bullet \tag{4.25}$$

$$\left(\left[t - t_{Begin\varphi}\right]^{\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1} + 1} \bullet \frac{1}{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) + 1} \right) + C$$

$$\{\mu_U(t)\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]^{\left(\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1}\right)} = \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet \left(\left[t - t_{Begin\varphi}\right]^{\frac{DRE_{\varphi} \bullet \{\mu_R\}_{j,\varphi}}{DRE_{\varphi} - 1} + 1} \right)}{\left(\frac{DRE_{\varphi} \bullet \{\mu_R(t)\}_{j,\varphi}}{DRE_{\varphi} - 1}\right) + 1} + C \tag{4.26}$$

$$\{\mu_U(t)\}_{j,\varphi} = \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet \lfloor t - t_{Begin\varphi} \rfloor}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1} + \frac{C}{\lfloor t - t_{Begin\varphi} \rfloor^{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right)}} \tag{4.27}$$

Solving for C:

$$\{\mu_U(t)\}_{j,\varphi} - \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet \lfloor t - t_{Begin\varphi} \rfloor}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1} = \frac{C}{\lfloor t - t_{Begin\varphi} \rfloor^{\left(\frac{DRE_\varphi \bullet \{\mu_R(t)\}_{j,\varphi}}{DRE_\varphi - 1}\right)}} \tag{4.28}$$

$$\lfloor t - t_{Begin\varphi} \rfloor^{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right)} \bullet \left[\{\mu_U(t)\}_{j,\varphi} - \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet \lfloor t - t_{Begin\varphi} \rfloor}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1}\right] = C \tag{4.29}$$

For t at the start of a phase:

$t = t_{Begin\varphi} - t_{Begin\varphi} = 0$

$$0 \bullet \left[\{\mu_U(t)\}_{j,\varphi} - \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet 0}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1}\right] = C \tag{4.30}$$

Thus:

$C = 0$

$$\{\mu_U(t)\}_{j,\varphi}^{\text{Predicted}} = \frac{\{\nu \bullet \mu_H\}_{j,\varphi} \bullet \lfloor t - t_{Begin\varphi} \rfloor}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1} \tag{4.31}$$

Equation 4.31 is the equation for the predicted value of $\{\mu_U(t)\}_{j,\varphi}$ for the Fault Introduction phases. Using equation 2.37:

$$\{\nu \bullet \mu_H\}_{j,\varphi} = \frac{DP_\varphi \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \bullet F_{j,\varphi}^{1-2SLI_\varphi} \tag{4.32}$$

Where:

$\{\nu \bullet \mu_H\}_{j,\varphi}$	unadjusted estimate of the fault introduction rate of the j-th fault categories in a phase φ
j	a category of faults introduced during phase j, j = RQ, DE or CO
φ	a lifecycle phase, ie., φ = RQ, RR, DE, DR, CO, CI, UT, IgT, ST, AT
$F_{j,\varphi}$	a constant
SLI_φ	Success Likelihood Index which varies between 0 (error is likely) and 1 (error is not likely)
DP_φ	fault potential per function point in phase φ
$fd_{j,\varphi}$	Fraction of faults of type j that originated in phase φ
$\bar{t}_{FP,\phi}$	mean effort necessary to develop a function point in phase φ

yields:

$$\{\mu_U(t)\}_{j,\varphi}^{\Pr edicted} = \frac{\frac{DP_\varphi \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \bullet F_{j,\varphi}^{1-2SLI_\varphi} \bullet \left[t - t_{Begin\varphi}\right]}{\left(\frac{DRE_\varphi \bullet \{\mu_R\}_{j,\varphi}}{DRE_\varphi - 1}\right) + 1} \tag{4.33}$$

Equation 4.14, the actual number of faults observed in a Fault Introduction phase, can be solved using equation 4.33 and equation 4.20.

$$\{\mu_U(t)\}_{j,\varphi \Pr edicted}^{Observed} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \{z_a\}_{j,\varphi} \left\{\mu_U(s)^{\Pr edicted}\right\}_{j,\varphi} ds \tag{4.14}$$

Substituting from equations 4.20 and 4.33:

$$\left\{\mu_U(t)\right\}^{Observed}_{j,\varphi \Pr edicted} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \frac{\frac{DRE_\varphi}{1-DRE_\varphi}}{\left[t-t_{Begin\varphi}\right]} \bullet \frac{\frac{DP_\varphi \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \bullet F_{j,\varphi}^{1-2SLI_\varphi} \bullet \left[t-t_{Begin\varphi}\right]}{\left(\frac{DRE_\varphi \bullet \left\{\mu_R\right\}_{j,\varphi}}{DRE_\varphi - 1}\right)+1} dt \qquad (4.34)$$

$$\left\{\mu_U(t)\right\}^{Observed}_{j,\varphi \Pr edicted} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \frac{DRE_\varphi}{1-DRE_\varphi} \bullet \frac{\frac{DP_\varphi \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \bullet F_{j,\varphi}^{1-2SLI_\varphi}}{\left(\frac{DRE_\varphi \bullet \left\{\mu_R\right\}_{j,\varphi}}{DRE_\varphi - 1}\right)+1} dt \qquad (4.35)$$

$$\left\{\mu_U(t)\right\}^{Observed}_{j,\varphi \Pr edicted} = \frac{DRE_\varphi}{1-DRE_\varphi} \bullet \left[\frac{DP_\varphi \bullet fd_{j,\varphi}}{\bar{t}_{FP,\varphi}} \bullet F_{j,\varphi}^{1-2SLI_\varphi}\right] \bullet$$

$$\left[\frac{1}{\left(\frac{DRE_\varphi \bullet \left\{\mu_R\right\}_{j,\varphi}}{DRE_\varphi - 1}\right)+1}\right] \bullet \left[t_{End\varphi} - t_{Begin\varphi}\right] \qquad (4.36)$$

The predicted number of faults observed in a Fault Introduction phase is obtained from Equation 4.36.

Next, the equation for the predicted number of faults observed in a Fault Removal phase is derived, using equation 4.14. The predicted value of μ_U, for Fault Removal phases, is derived from Equation 2.6. For Fault Removal phases, the Fault Introduction component is zero:

$$\left\{\nu(t)\mu_H(t)\right\}_{j,\varphi} = 0$$

Leaving:

$$\frac{d\mu_U(t)_{j,\varphi}}{dt} = \left\{z_a(t)\right\}_{j,\varphi}\left\{\mu_R(t)\right\}_{j,\varphi}\left\{\mu_U(t)\right\}_{j,\varphi} \qquad (4.37)$$

Simplifying:

$$\frac{d\mu_U(t)_{j,\varphi}}{\{\mu_U(t)\}_{j,\varphi}} = \left[\{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}\right] dt \tag{4.38}$$

$$\int \frac{1}{\{\mu_U(t)\}_{j,\varphi}} d\mu_U(t)_{j,\varphi} = \int \left[\{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}\right] dt \tag{4.39}$$

Integrating and since $\{z_a(t)\}_{j,\varphi}\{\mu_R(t)\}_{j,\varphi}$ is constant over phase φ:

$$\ln\left(\{\mu_U(t)\}_{j,\varphi}\right) = \left[\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]\right] + C \tag{4.40}$$

With initial conditions, for t at the start of a phase:

$$t = t_{Begin\varphi} - t_{Begin\varphi} = 0$$

$$C = \ln\left(\{\mu_U(t)\}_{j,\varphi}\right) + \left[-\left(\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet 0\right)\right] \tag{4.41}$$

Equation 4.40 becomes:

$$\{\mu_U(t)\}_{j,\varphi} = e^{\left[\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]\right] + C} \tag{4.42}$$

Substituting for C:

$$\{\mu_U(t)\}^{\Pr edicted}_{j,\varphi} = e^{\left[\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]\right]} \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}^{\Pr edicted}_{j,\varphi-1} \tag{4.43}$$

Equation 4.14, the predicted number of faults observed in a Fault Removal phase, can be solved using equation 4.43 and equation 4.13.

$$\{\mu_U(t)\}^{Observed}_{j,\varphi\,\Pr edicted} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \{z_a\}_{j,\varphi} \left\{\mu_U(s)^{\Pr edicted}\right\}_{j,\varphi} ds \tag{4.14}$$

$$\{\mu_U(t)\}^{Observed}_{j,\varphi\,\Pr edicted} = \int_{t_{Begin\varphi}}^{t_{End\varphi}} \{z_a\}_{j,\varphi} \bullet e^{\left[\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet \left[t - t_{Begin\varphi}\right]\right]} \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}^{\Pr edicted}_{j,\varphi-1} dt \tag{4.44}$$

$$\{\mu_U(t)\}^{Observed}_{j,\varphi\,\Pr edicted} = \{z_a\}_{j,\varphi} \bullet e^{\left[-\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet \left[t_{Begin\varphi}\right]\right]} \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}^{\Pr edicted}_{j,\varphi-1} \int_{t_{Begin\varphi}}^{t_{End\varphi}} e^{\left[-\{z_a\}_{j,\varphi}\{\mu_R\}_{j,\varphi} \bullet [t]\right]} dt$$

(4.45)

$$\left\{\mu_U(t)\right\}_{j,\varphi \Pr edicted}^{Observed} = \left\{z_a\right\}_{j,\varphi} \bullet e^{\left[-\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{Begin\varphi}\right]\right]} \bullet \frac{1}{\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}} \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}_{j,\varphi-1}^{\Pr edicted} \bullet$$

$$\left[e^{\left[-\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet[t]\right]}\right]_{t_{Begin\varphi}}^{t_{End\varphi}} \quad (4.46)$$

$$\left\{\mu_U(t)\right\}_{j,\varphi \Pr edicted}^{Observed} = \frac{1}{\left\{\mu_R\right\}_{j,\varphi}} \bullet e^{\left[-\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{Begin\varphi}\right]\right]} \bullet$$

$$\left[e^{\left[\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{End\varphi}\right]\right]} - e^{\left[\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{Begin\varphi}\right]\right]}\right] \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}_{j,\varphi-1}^{\Pr edicted} \quad (4.47)$$

$$\left\{\mu_U(t)\right\}_{j,\varphi \Pr edicted}^{Observed} = \frac{1}{\left\{\mu_R\right\}_{j,\varphi}} \bullet \left[e^{\left[\left\{z_a\right\}_{j,\varphi}\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{End\varphi}-t_{Begin\varphi}\right]\right]} - 1\right] \bullet \left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}_{j,\varphi-1}^{\Pr edicted} \quad (4.48)$$

Replacing by the expression of $\{z_a\}_{j,\varphi}$ (see equation (4.13)) we obtain:

$$\left\{\mu_U(t)\right\}_{j,\varphi \Pr edicted}^{Observed} = \frac{1}{\left\{\mu_R\right\}_{j,\varphi}} \bullet \left[e^{\left[\left\{\overline{z_a}\right\}_{j,\varphi}+\left(SLI_\varphi-0.5\right)\left[\left\{z_{aMAX}\right\}_{j,\varphi}-\left\{z_{aMIN}\right\}_{j,\varphi}\right]\left\{\mu_R\right\}_{j,\varphi}\bullet\left[t_{End\varphi}-t_{Begin\varphi}\right]\right]} - 1\right] \bullet$$

$$\left\{\mu_U\left(t_{End(\varphi-1)}\right)\right\}_{j,\varphi-1}^{\Pr edicted} \quad (4.49)$$

The predicted number of faults observed in a Fault Removal phase is obtained from Equation 4.49. Using these equations, along with a known value for the actual, observed faults in a Fault Removal phase the value for SLI can be optimized.

4.2.2 Optimization Approaches

This research investigates the use of a software application to choose the optimal SLI values for the Fault Introduction – Fault Removal pair of lifecycle phases. The algorithm applied to solve for SLI is to systematically step through the range of SLI values, 0 to 1, and choose the combination of parameters that most accurately portrays the SLI values obtained through expert opinion.

To perform the systematic search for optimal SLI, the software application directs the user to enter the required parameters of the software reliability model, for both of the phases. Equations 4.36 and 4.49 are used in the software application. The parameters, which have been entered by the user, are used to calculate first the predicted faults observed for the Fault Introduction phase as the SLI for that phase increments in 0.01 steps from 0 to 1. At each step the corresponding predicted faults observed for the Fault Removal phase is calculating by stepping through its SLI in 0.01 steps from 0 to 1. The result of this algorithm is a table of 10,000 data sets. The data sets contain the SLI of each phase, Fault Introduction and Fault Removal, the predicted faults in the Fault Introduction phase, the predicted faults in the Fault Removal phase, the predicted faults to be observed and the difference between the actual and predicted faults observed in the Fault Removal phase for each of the 10,000 entries. The data set is sorted based on the difference between predicted and actual faults observed in the Fault Removal phase. The algorithm calls for the user to choose the highest ranked data set, in terms of difference, which is closest in SLI values to the SLIs obtained from expert opinion. In applying this algorithm a data set of SLI values for Fault Introduction and Fault Removal phases has been found within the top 5 sets of results for each Fault Introduction (FI) – Fault Removal (FR) phase implemented. This approach is called the FI-FR SLI Optimization by SLI Fitting method.

This research discusses three types of faults:

- Number of faults predicted
- Number of faults predicted to be observed

- Number of faults actually observed

The number of faults predicted is the number of faults a model calculates will occur in a phase of the software development. The number of faults predicted to be observed is the number of faults that a fault removal phase predicts will be removed, as calculated by a model. This is the number of faults at the end of a fault removal phase minus the number of faults at the start of a fault removal phase. The third type of fault is the number of faults actually observed. This is the number of faults observed by the project team in a fault removal phase.

This method uses extensively the results of the analysis done in section 4.2.1.

4.2.3 Project Data with SLI Optimization

The software reliability model was applied with the SLI values obtained from the FI-FR SLI Optimization by SLI Fitting method. The method was applied to the Waterfall and the FDD lifecycle projects.

The Waterfall and FDD project data include the value of requirements, design, coding and cumulative faults at each phase in the development cycle, as in the two other applications of the reliability model. In review, the first application of the model to the Waterfall and FDD projects had only SLI values in the Fault Introduction phases. The second application was enhanced to allow the entry of SLI values in Fault Removal phases. This application of the model, a third approach, makes use of the FI-FR SLI Optimization by SLI Fitting method to obtain SLI values for input to the model.

For the Waterfall lifecycle application of the FI-FR SLI Optimization by SLI Fitting method yields the following data. The test case for this Waterfall lifecycle is test case (#39) and the parameters are given in Appendix D.

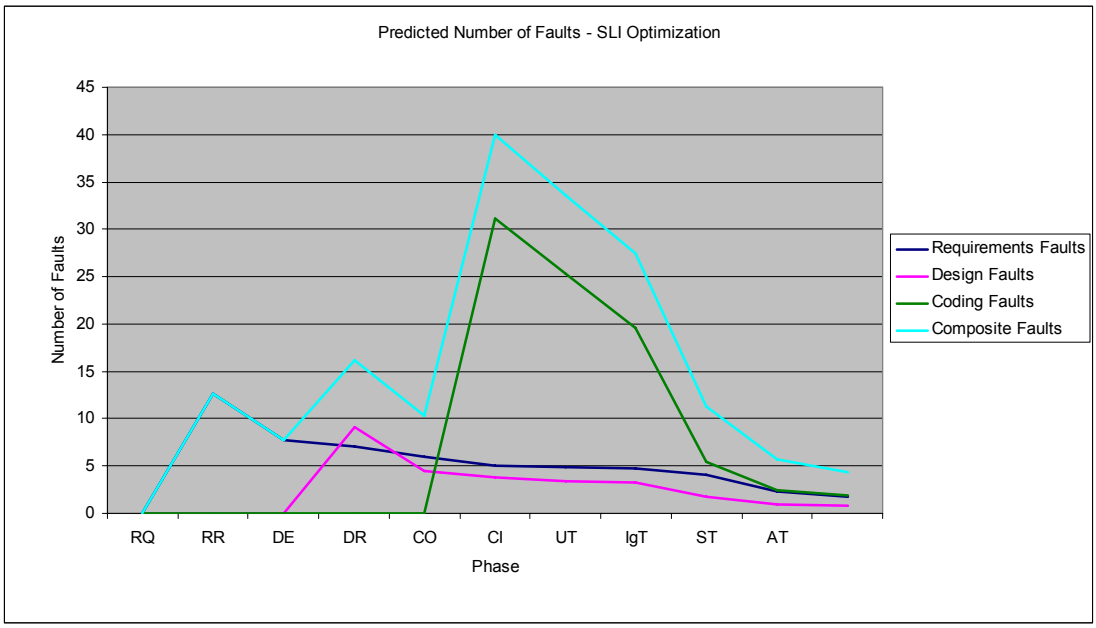


Figure 25 Model Predicted Fault Count – Waterfall Project – SLI Optimized in FI-FR Phase Pairs

The reliability model, with the parameters as above, expects that 4.38 predicted faults could be expected in the system at the end of development. The percent error between the expected observed and actually observed faults at 5% showed a substantial improvement to the previous implementations of 41% and 24%. The phases in which SLI value optimization was performed, RQ-RR and DE-DR showed the ability to track the model to actual data. The SLI value for the ST phase could not be optimized due to the constraint of having the Fault Introduction phase

immediately prior to the Fault Removal phase having the SLI value optimized. Further analysis is performed in section 4.2.4.

Table 35 Model Expected Observed Faults Vs. Observed Faults – Waterfall Lifecycle – SLI Optimized in FI-FR Phase Pairs

Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phases Expected Faults	SLI Optimized Expected Faults	Observed & Repaired Faults
RR	RQ Faults	4.05	4.20	6.01	6
DR	DE Faults	5.54	5.22	6.96	7
ST	Cumulative Faults	2.26	5.69	7.95	7
Total		11.85	15.11	20.92	20
Percent Error (Expected vs. Observed Faults)		40.75	24.45	4.6	

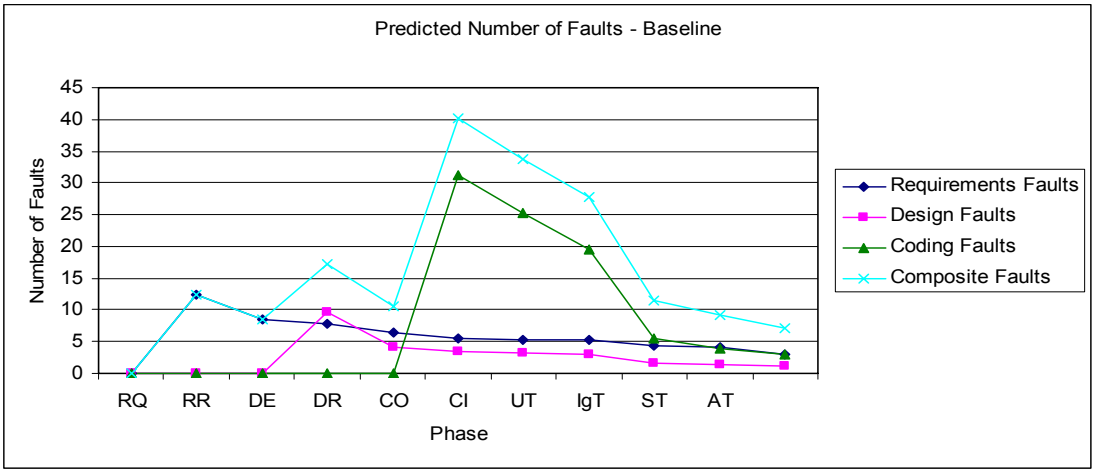
Table 36 Model Expected Observed Faults Vs. Observed Faults –Waterfall Project – SLI Optimized in FI-FR Phase Pairs (Without ST Phase)

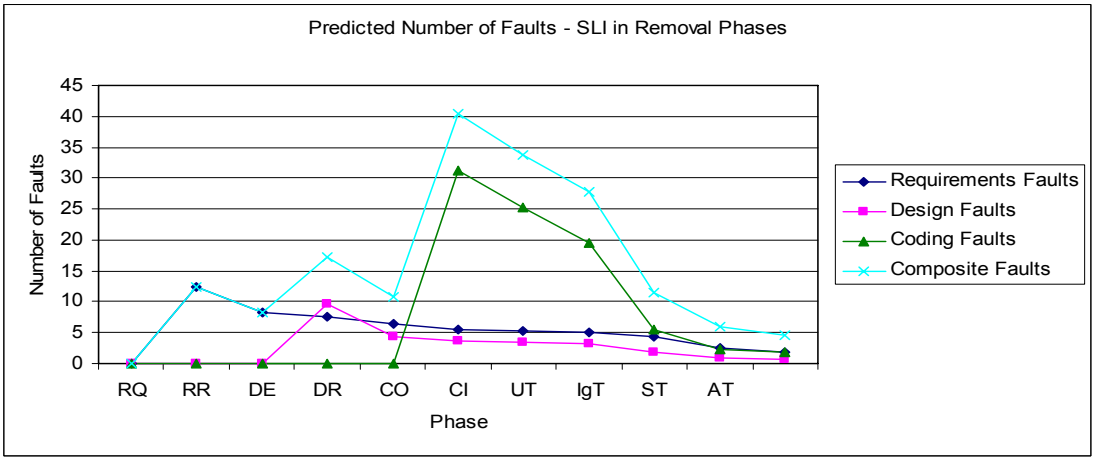
Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phases Expected Faults	SLI Optimized Expected Faults	Observed & Repaired Faults
RR	RQ Faults	4.05	4.20	6.01	6
DR	DE Faults	5.54	5.22	6.96	7
Total		9.59	9.42	12.97	13
Percent Error (Expected vs. Observed Faults)		26.23	27.54	0.23	

In Table 36, the expected observed and observed fault data is shown without the ST phase. This phase does not have an immediate Fault Introduction phase as a predecessor phase. Thus, the ST phase could not be optimized, in the current research. Extending the optimization beyond one preceding phase is presented as a

further research item. The percent error is improved using only the optimized Fault Introduction and Removal phases (RQ-RR, DE-DR) as the phases to compare. The percent error, without ST, for the SLI Optimized model reduces to 0.23%, compared to 26% and 28%, for the Baseline and SLI in Removal Phases models

A side by side comparison of the chart of predicted fault count from the Baseline model, the predicted fault count from the SLI in Removal Phase model and the predicted fault count from the SLI Optimization model follows. The comparison shows slight variations in the phases as a result of using the SLI optimization. These variations are in the RQ-RR and DE-DR phases and transitions as the SLI optimization attempts to control the model. The overall predicted faults left in the system also dropped from 7.06 (Baseline) to 4.48 (SLI in Removal Phases) back to 4.38 (SLI Optimized) for this model. The table of SLI values, Table 37, illustrates how this model departs from the obtained SLI through expert opinion values and uses the values obtained from optimization value for the RQ, RR, DE and DR phases. Also shown is how there is little deviation from the two SLI values (expert opinion and optimized) for the SLI Optimized model.





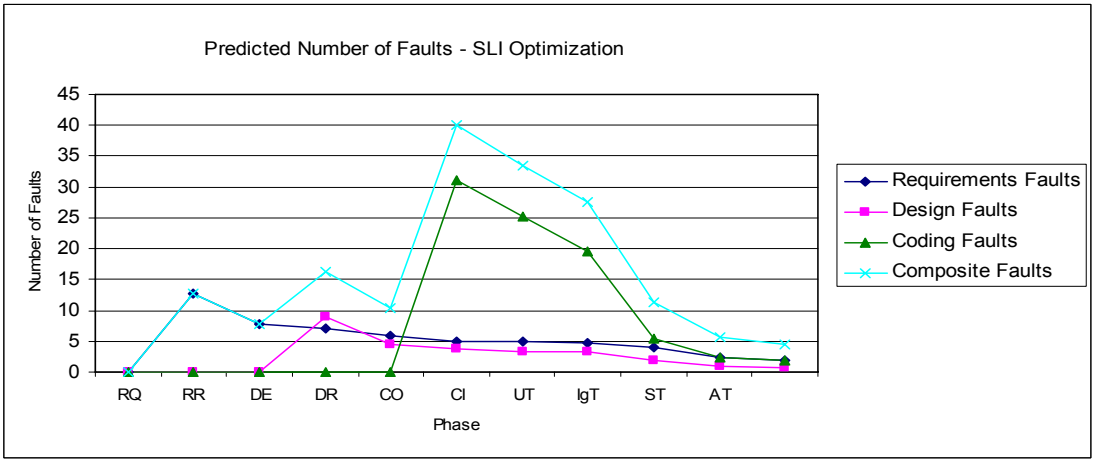


Figure 26 Comparison of Predicted Number of Faults – Baseline, SLI in Removal Phases and SLI Optimization Models– Waterfall Project

Table 37 SLI Values – Baseline, SLI in Removal Phases and SLI Optimization Models and Expert Opinion – Waterfall Project

Activity	SLI - Baseline	SLI – In Removal Phases	SLI - Optimized	SLI from Expertise
RQ	0.213	0.213	0.21	0.213
RR	No SLI	0.6623	0.65	0.6623
DE	0.413	0.413	0.43	0.413
DR	No SLI	0.5361	0.53	0.5361
CO	0.383	0.383	0.383	0.383
CI	No SLI	No SLI	No SLI	Not Done
UT	No SLI	No SLI	No SLI	Not Done
IgT	No SLI	No SLI	No SLI	Not Done
ST	No SLI	0.675	0.675	0.675
AT	No SLI	No SLI	No SLI	Not Done

For the FDD lifecycle application of the FI-FR SLI Optimization by SLI Fitting method yields the following data. The test case for this FDD lifecycle is test case (#29) and the parameters are given in Appendix D.

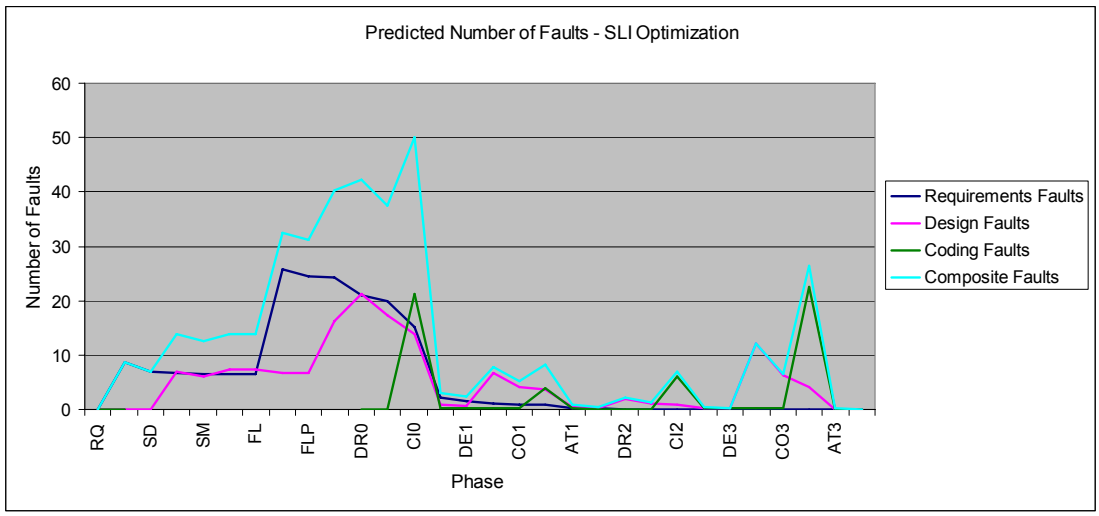


Figure 27 Model Predicted Fault Count – FDD Project – SLI Optimized in FI-FR Phase Pairs

Table 38 Model Expected Observed Faults Vs. Observed Faults –FDD Project – SLI Optimized in FI-FR Phase Pairs

Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phases Expected Faults	SLI Optimized Expected Faults	Observed & Repaired Faults
RR	RQ Faults	2.62	2.12	2.000	2
SDR	DE Faults	1.26	0.93	1.000	1
SMR	DE Faults	0.08	0.06	0.078	0
FLR	RQ Faults	0.79	0.62	1.333	1
DR0	DE Faults	5.21	4.04	4.037	4
ST0	Cumulative Faults	2.75	9.60	15.600	8
DR1	DE Faults	2.80	1.80	3.280	3
ST1	Cumulative Faults	0.36	0.78	1.420	4
DR2	DE Faults	2.78	2.17	1.170	1
ST2	Cumulative Faults	0.18	0.48	0.448	1
DR3	DE Faults	6.78	5.46	8.081	8
CI3	CO Faults	26.46	19.81	21.142	21
ST3	Cumulative Faults	0.01	0.11	0.157	1
Total		52.08	47.98	59.746	55
Percent Error (Expected vs. Observed Faults)		5.31	12.76	8.629	

The reliability model, with the parameters as above, predicts that 0.098 faults could be expected in the system at the end of development. The percent error between the expected observed and actual observed faults at 9% showed an improvement to the previous implementation of 12% and similar results to the baseline implementation of 5%.

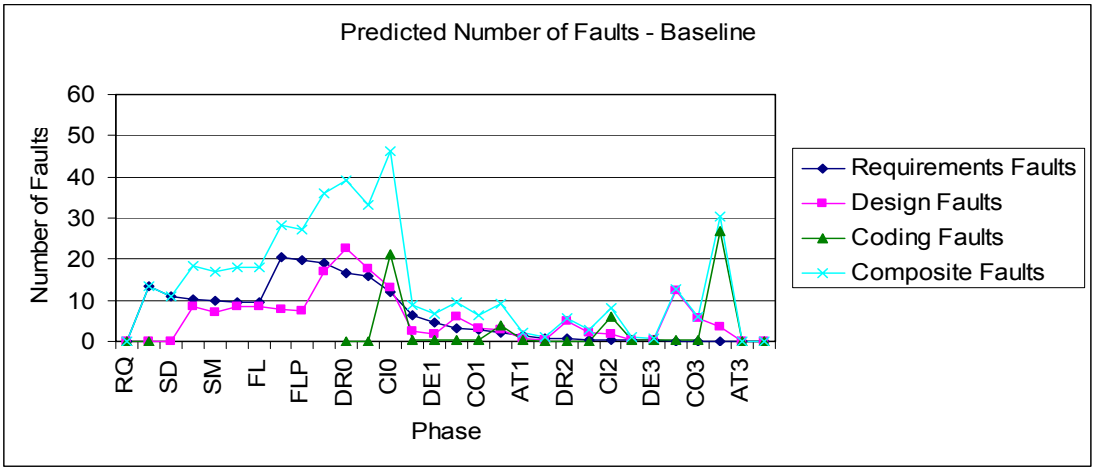
Table 39 Model Expected Observed Faults Vs. Observed Faults –FDD Project – SLI Optimized in FI-FR Phase Pairs (Without ST Phases)

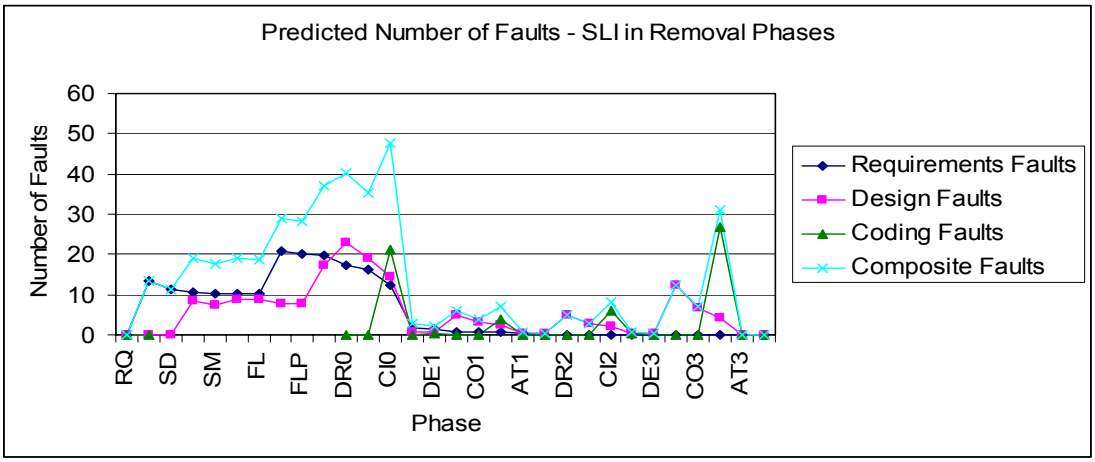
Inspection Phase	Fault Type	Baseline Model Expected Faults	SLI in Removal Phases Expected Faults	SLI Optimized Expected Faults	Observed & Repaired Faults
RR	RQ Faults	2.62	2.12	2.000	2
SDR	DE Faults	1.26	0.93	1.000	1
SMR	DE Faults	0.08	0.06	0.078	0
FLR	RQ Faults	0.79	0.62	1.333	1
DR0	DE Faults	5.21	4.04	4.037	4
DR1	DE Faults	2.80	1.80	3.280	3
DR2	DE Faults	2.78	2.17	1.170	1
DR3	DE Faults	6.78	5.46	8.081	8
CI3	CO Faults	26.46	19.81	21.142	21
Total		48.78	37.01	42.121	41
Percent Error (Expected vs. Observed Faults)		18.98	9.73	2.734	

In Table 39, the expected observed and actual observed fault data is shown without the ST phases. These ST phases do not have an immediate Fault Introduction phase as a predecessor phase. Thus, the ST phases could not be optimized, in the current research. Extending the optimization beyond one preceding phase is presented as a further research item. The percent error is improved using only the optimized Fault Introduction and Removal phases (RQ-RR, SD-SDR, SM-SMR, DEn-DRn, CO3-CI3) as the phase to compare. The percent error, without ST, for the SLI Optimized model reduces to 2.7%, compared to 19% and 10%, for the Baseline and SLI in Removal Phases models

A side by side comparison of the chart of predicted fault count from the Baseline model, the predicted fault count from the SLI in Removal Phase model and

the predicted fault count from the SLI Optimization model follows. The comparison shows slight variations in the phases as a result of using the SLI optimization. These variations are in the RQ-RR and DE-DR phases and transitions as the SLI optimization attempts to control the model. The overall predicted faults left in the system also stayed similar from 0.063 (Baseline) to 0.095 (SLI in Removal Phases) back to 0.098 (SLI Optimized) for this model. The table of SLI values, Table 40, illustrates how this model departs from the obtained SLI through expert opinion values and uses the values obtained from optimization value for the RQ, RR,SD, SDR, SM, SMR, DEn,DRn, CO3 and CI3 phases. Also shown is how there is little deviation, with three possible exceptions – DR0, DE2 and DR2, from the two SLI values (expert opinion and optimized) for the SLI Optimized model.





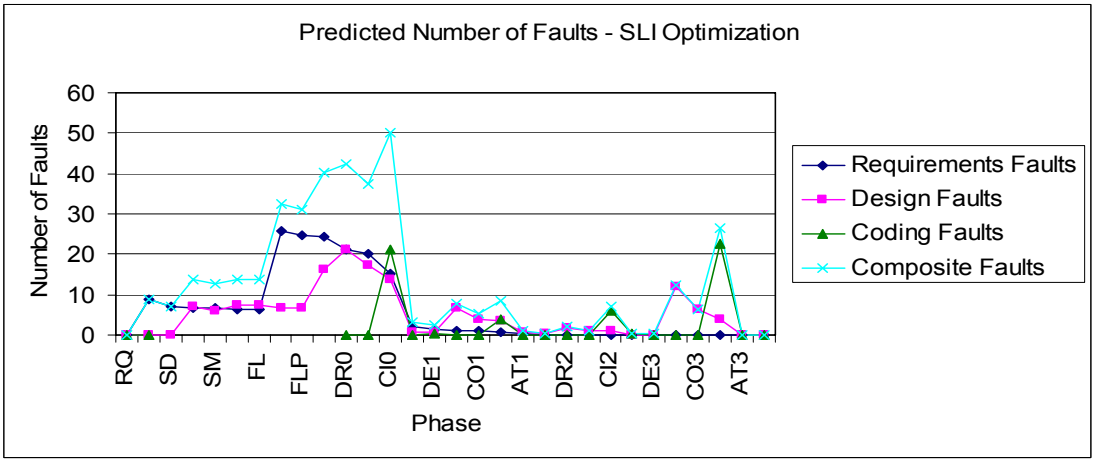


Figure 28 Comparison of Predicted Number of Faults – Baseline, SLI in Removal Phases and SLI Optimization Models– FDD

Table 40 SLI Values – Baseline, SLI in Removal Phases and SLI Optimization Models and Expert Opinion – FDD Project

Activity	SLI - Baseline	SLI – In Removal Phase	SLI - Optimized	SLI from Expertise
RQ	0.4259	0.4259	0.56	0.4259
RR	No SLI	0.3852	0.41	0.3852
SD	0.7093	0.7093	0.77	0.7093
SDR	No SLI	0.3889	0.28	0.3889
SM	0.6463	0.6463	0.64	0.6463
SMR	No SLI	0.3093	0.33	0.3093
FL	0.3889	0.3889	0.21	0.3889
FLR	No SLI	0.363	0.54	0.363
FLP	0.3704	0.3704	0.3704	0.3704
DE0	0.6778	0.6778	0.71	0.6778
DR0	No SLI	0.3389	0.12	0.3389
CO0	0.6593	0.6593	0.6593	0.6593
CI0	No SLI	No SLI	No SLI	0.3981
UT0	No SLI	No SLI	No SLI	Not Done
IgT0	No SLI	No SLI	No SLI	Not Done
ST0	No SLI	0.6556	0.6556	0.6556
AT0	No SLI	No SLI	No SLI	Not Done
DE1	0.6778	0.6778	0.59	0.6778
DR1	No SLI	0.3389	0.28	0.3389
CO1	0.6593	0.6593	0.6593	0.6593
CI1	No SLI	No SLI	No SLI	0.3981
UT1	No SLI	No SLI	No SLI	Not Done
IgT1	No SLI	No SLI	No SLI	Not Done
ST1	No SLI	0.6556	0.6556	0.6556
AT1	No SLI	No SLI	No SLI	Not Done
DE2	0.6778	0.6778	0.95	0.6778
DR2	No SLI	0.3389	0.28	0.3389
CO2	0.6593	0.6593	0.6593	0.6593
CI2	No SLI	No SLI	No SLI	0.3981
UT2	No SLI	No SLI	No SLI	Not Done
IgT2	No SLI	No SLI	No SLI	Not Done
ST2	No SLI	0.6556	0.6556	0.6556
AT2	No SLI	No SLI	No SLI	Not Done
DE3	0.6778	0.6778	0.68	0.6778
DR3	No SLI	0.3389	0.41	0.3389
CO3	0.6593	0.6593	0.71	0.6593
CI3	No SLI	0.3981	0.26	0.3981
UT3	No SLI	No SLI	No SLI	Not Done
IgT3	No SLI	No SLI	No SLI	Not Done
ST3	No SLI	0.6556	0.6556	0.6556
AT3	No SLI	No SLI	No SLI	Not Done

The next section provides for comparison between the predicted to be observed fault data of the software reliability model for the base, enhanced SLI and optimized SLI versions and the actual observed fault data.

4.2.4 Analysis of Predicted to Observed Fault Data

The objective in analyzing the predicted to be observed fault data from the various versions of the software reliability model and the actual observed fault data is to determine which model version best predicts the actual observed data. Another objective is to demonstrate the ability of the model to maintain control of the software process through optimization of the SLI parameter.

Schunn and Wallach [33] discuss the issues involved in how to use goodness-of-fit to evaluate models. There are common problems associated with goodness-of-fit. These include free parameters, noise in data, overfitting and uninteresting inflations of goodness-of-fit parameters. There are two types of numerical measures of goodness-of-fit. These are measures of deviation from exact data location and measures of how well the trend magnitudes are captured. This research is interested in measures of deviation. Schunn and Wallach [33] recommend a combination of r^2 for trend relative magnitude and RMSD (root mean squared deviation) or RMSSD (root mean squared scaled deviation) for deviation from exact data location.

The r and r^2 measure provide a sense of the overall quantitative variance in relative sizes and provide a scale (0 to 1) to indicate how well a model is doing [33].

Values close to 1 indicate a strong linear relationship and values close to 0 indicate a weak linear relationship.

The most popular measures of goodness-of-fit to exact location are the mean squared deviation (MSD) and its square root, RMSD [33]. The MSD is defined as [33]:

$$MSD = \frac{\sum_{i=1}^{k} (m_i - d_i)^2}{k}$$

Where, m_i is the model mean for each point i, d_i is the data mean for each point i and k is the number of points i being compared. The value of RMSD is equal to the square root of MSD. The RMSD measure is versatile and well-used in the research community making it a strong choice in goodness-of-fit evaluations.

4.2.4.1 Waterfall Project

In Table 41 the fault data is presented for a side by side goodness-of-fit comparison. The RMSD measure is the better goodness-of-fit measure for this data set.

Table 41 Analysis of Model Results for Waterfall Project

Phase	Predicted to be Observed Faults - Baseline	Predicted to be Observed Faults – SLI in Removal Phases	Predicted to be Observed Faults – SLI Optimization	Actual Observed Faults
RR	4.05	4.2	6.006	6
DR	5.54	5.22	6.960	7
ST	2.26	5.69	7.948	7
Total	**11.85**	**15.11**	**20.914**	**20**
r^2	0.003	0.905	0.741	
RMSD	3.0769	1.6457	0.5480	

The chart in Figure 29 displays the plot of the predicted to be observed faults from the software reliability model of waterfall project in its baseline, enhancement (SLI in Removal Phases) and optimized (SLI Optimized) forms. The actual observed faults are also plotted. These plots follow a similar path from the RR phase to the end of the lifecycle. The chart indicates that visually the predicted to be observed faults of the model when the SLI is optimized maps most closely to the actual observed faults.

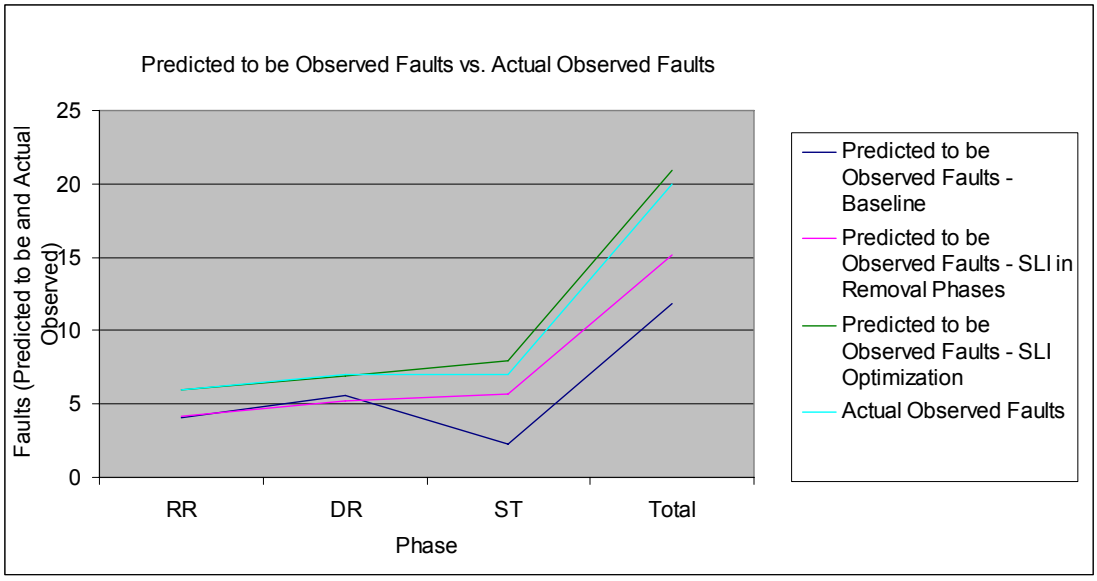


Figure 29 Model Predicted to be Observed Faults vs. Actual Observed Fault Data for Waterfall Project

In the analysis above the SLI optimized predicted to be observed faults results are most closely related to the actual observed faults. This is the first outcome of the analysis.

The outcome to show control of the model through the SLI parameter is also portrayed in the optimized data set. By optimizing the SLI of the Fault Introduction – Fault Removal phase pairs with actual observed fault data the model has been shown

to map closely to the predicted to be observed faults. In Table 37 the SLI used for each phase of the three models is shown. The table shows that SLI in the SLI optimized model is consistent with the SLI derived from expert opinion. The SLIs in the RQ, RR, DE and DR phases are used in the Fault Introduction – Fault Removal phases. An entry of "No SLI" indicates that the removal phase did not incorporate the SLI into the model for that removal phase. An entry of "Not Done" indicates that expert opinion for that removal phase was not calculated as an SLI value.

4.2.4.2 FDD Project

The FDD project presents an analysis of the fault data for a side by side qualitative comparison. In Table 42 the fault data is presented for a side by side goodness-of-fit comparison. The RMSD measure is the better goodness-of-fit measure for this data set. The value of predicted to be observed faults for ST0 in the SLI Optimization model has a negative impact on the RMSD for this model.

Table 42 Analysis of Model Results for FDD Project

Phase	Predicted to be Observed Faults - Baseline	Predicted to be Observed Faults – SLI in Removal Phases	Predicted to be Observed Faults – SLI Optimization	Actual Observed Faults
RR	2.62	2.12	2.000	2
SDR	1.26	0.93	1.000	1
SMR	0.08	0.06	0.078	0
FLR	0.79	0.62	1.333	1
DR0	5.21	4.04	4.037	4
ST0	2.75	9.6	15.600	8
DR1	2.8	1.8	3.280	3
ST1	0.36	0.78	1.420	4
DR2	2.78	2.17	1.170	1
ST2	0.18	0.48	0.448	1
DR3	6.78	5.46	8.081	8
CI3	26.46	19.81	21.142	21
ST3	0.01	0.11	0.157	1
Total	**52.08**	**47.98**	**59.746**	**55**
r^2	0.888	0.946	0.882	
RMSD	2.4642	1.3822	2.2478	

The chart in Figure 30 displays the plot of the predicted to be observed faults from the software reliability model of the FDD project in its baseline, enhancement (SLI in Removal Phases) and optimized (SLI Optimized) forms. The actual observed faults are also plotted. These plots follow a similar path from the RR phase to the end of the lifecycle. The chart indicates that visually the predicted to be observed faults of the model when the SLI is optimized maps most closely to the actual observed

faults. This chart contains more data points than the Waterfall lifecycle. The iterations which contain more function points, iteration 0 and iteration 3, are clearly visible.

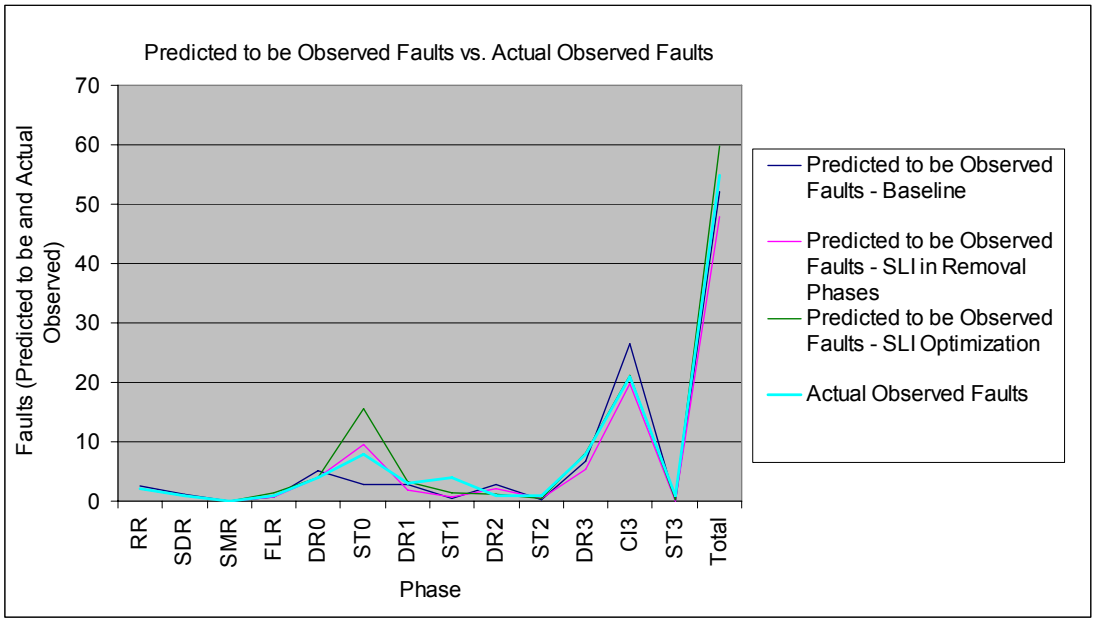


Figure 30 Model Predicted to be Observed Faults vs. Actual Observed Fault Data for for FDD Project

In the analysis above the SLI optimized predicted to be observed faults results are most closely related to the actual observed faults, although the predicted to be observed fault data for ST0 of the SLI Optimization model skews the data set. This is the first outcome of the analysis.

The other outcome, to show control of the model through the SLI parameter, is also portrayed in the data set from the SLI Optimization model. By optimizing the SLI of the Fault Introduction – Fault Removal phase pairs with actual observed fault data the model has been shown to map closely to the actual observed faults. There is

one outlier associated with the optimized data set, ST0, from Table 42. The other model variations have multiple outliers.

In Table 40, the SLI used for each phase of the three model iterations is shown. The table shows that SLI in the optimized model is consistent with the SLI derived from expert opinion. The SLIs, that are optimized, are in the RQ, RR, SD, SDR, SM, SMR, FL, FLR, DE0, DR0, DE1, DR1, DE2, DR2, DE3, DR3, CO3 and CI3 phases which are the Fault Introduction – Fault Removal phases. In these phase pairs there exists actual data from the removal phase to use in the equations for optimization. An entry of "No SLI" indicates that the removal phase did not incorporate the SLI into the model for that removal phase. An entry of "Not Done" indicates that expert opinion for that removal phase was not calculated as an SLI value.

Another comparison uses the value of the predicted number of faults left in the system at the end of the respective lifecycle. Table 43 displays the results for this value for each project type, Waterfall and FDD.

Table 43 Model Predicted Fault Count at Lifecycle End – Waterfall and FDD

Project Type	Lifecycle Type	
	Waterfall	**FDD**
Base	7.06	0.063
Enhancement	4.48	0.095
Optimized	4.38	0.098

The data in the table illustrate that a trend is not indicated, related to predicted fault count at the end of a lifecycle, for enhancements to the base model. For the Waterfall project, the model predicted fault count is reduced by improvements to the model. The FDD project the model predicted fault count at lifecycle end is so small that these changes are not significant. The small value of the model predicted fault count for the FDD project indicates that the iterative process may affect the predicted fault count by iterating through multiple review and inspection phases. With more removal phases than introduction phases per iteration, the process of reducing the fault count occurs naturally. The result of this analysis leads to the observation that model improvements increase the accuracy of phase by phase metrics but have an inconclusive effect on lifecycle end predicted to be observed fault count.

Chapter 5: Software Tools for Model Analysis

5.1 Waterfall Project Model Analysis Tool

Figure 31 Waterfall Project Analysis Tool Graphical User Interface

The Waterfall Project Model Analysis Tool was designed to automate the calculations performed to obtain predicted fault data from the software reliability model. The tool was developed using the C# language within Microsoft Visual Studio 2008©. The tool allows the user to enter the parameters of the Fault Introduction and Fault Removal phases. The entering of the parameters is done by reading the parameters from an ASCII formatted data file. Once data is input to the tool, the tool shows the value of each computed parameter, the phase start and stop time and the number of faults present in the system and the end of each phase on the graphical user interface (GUI). The predicted fault data is shown as requirements,

design, coding and cumulative fault values for each of the phases. The actual observed and predicted fault values are displayed. Various combinations of parameter input values are entered as a test case number, for historical purposes and to re-run prior tests. The tool has the capability to print a report of the tool results for a test case.

5.2 FDD Process Model Analysis Tool

Figure 32 FDD Project Analysis Tool Graphical User Interface

The FDD Project Model Analysis Tool was designed to automate the calculations performed to obtain predicted fault data from the software reliability model. The tool was developed using the C# language within Microsoft Visual Studio 2008©. The tool has all the capabilities of the Waterfall Project Analysis

Tool. This tool allows for the extra phases inherent in the iterative lifecycle of the FDD project.

5.3 SLI Optimization Tool

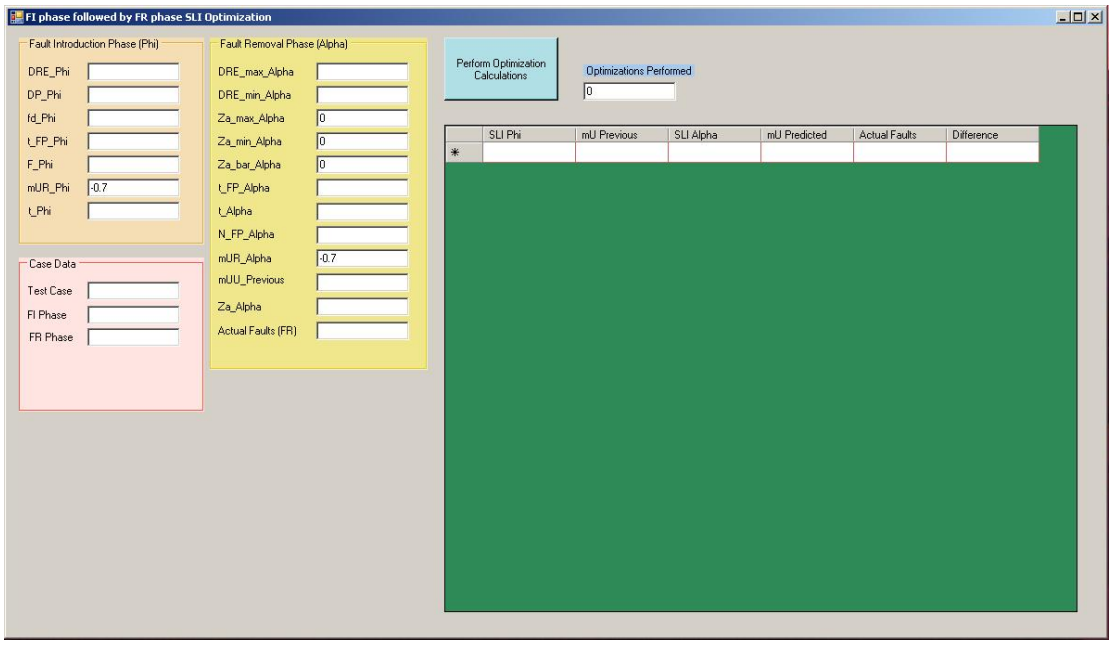


Figure 33 SLI Optimization Tool Graphical User Interface

The SLI Optimization Tool was designed to automate the calculations performed to obtain optimized SLI values for a Fault Introduction – Fault Removal pair of phases in a lifecycle to be used with the software reliability model. The tool was developed using the C# language within Microsoft Visual Studio 2008©.

In section 4.2.2 the use of a software application to choose the optimal SLI values for the Fault Introduction – Fault Removal pair of lifecycle phases is discussed and an algorithm is introduced The tool programmatically solves the equation for actual observed faults in a removal phase. The algorithm solves for SLI by systematically stepping through the range of SLI values, 0 to 1. These combinations

are displayed to the user on a GUI in the form of a table. The table may be sorted in descending or ascending order for any column of the parameter table. The parameters are Fault Introduction SLI, Fault Removal SLI, predicted to be observed fault count of the previous (Fault Introduction) phase, predicted to be observed fault count of the current (Fault Removal phase), actual observed faults in the Fault Removal phase and the difference between the actual observed faults and the predicted to be observed faults in the Fault Removal phase. Currently, the table will have 10,000 entries. The parameters required by the tool are entered manually into the GUI. The user will browse the list and choose the SLI pair which meets his criteria. The criteria that have been selected for this research selects an SLI pair which is in the top 50 of the table entries and has a close value to the SLI values for the pair given by expert opinion.

Chapter 6: Conclusions and Further Research

6.1 Conclusions

This research applied a theoretical software reliability model to two distinct software projects. The software projects were developed using different lifecycles. These lifecycles were the Waterfall lifecycle and the Feature Driven Development lifecycle adapted from an Agile approach. To apply the reliability model to the FDD lifecycle, extensions were made. The research obtained predicted fault data and made comparisons to actual observed fault data. The reliability model was modified and improved upon by creating methodologies for controlling and enhancing the accuracy of the model through the SLI parameters.

The conclusions reached in this study are presented in the following discussions.

The implementation of the software reliability model [12] to industry projects requires knowledge of the domains, model and application. The practitioner will be required to make decisions about the project regarding the metrics, effort (in terms of function point counts), fault data and lifecycle. The practitioner will also be required to adapt the model to various nuances of the model such as; application of SLI, optimization schemes and lifecycle phase determination among others in order for implementation to be successful. With the help of the tools developed in this research, the practitioner needs to know little of the underlying architecture and theory of how the model is constructed to be able to obtain model reliability statistics.

Another conclusion made refers to one of the contributions of this research. This is the tailoring of the Function Point Analysis to transition to an Agile development lifecycle. One can conclude that it is possible to adapt the function point counting to other various lifecycles by extending or deleting function point counting rules and validating these modifications. At this point in software development there are many lifecycles available.

The use of defect severity in classifying the faults detected by observation is important. The model is implemented to account for high severity (Category 1 and 2) faults. Thus, the observed faults need to be classified.

An inventory of the model parameters and metrics indicated which are obtained from industry standards, which are calculated from industry standards and which are gathered from a non-industry source (such as expert opinion). The parameters based on industry metrics are not able to be optimized. They are based on years of industry research and placed in various bins based on project type and size. The parameters which are calculated from industry metrics are also not able to be optimized. Parameters that may be optimized include those based on or partially based on non-industry sources. For example, defect potential per function point is calculated from type and size of the software project and from the interpolation method to use for determining the number of function points for a project of this type. The type and size is obtained from industry metrics and the interpolation method is derived. The value may be optimized. The third type of parameter is one derived without industry metrics, such as SLI. The SLI value is based on expert opinion and able to be optimized.

The variability of certain parameters is important. There are two separate ways to use the staff hours term in the model. As discussed above, this metric is both calculated and based on actual reported data. The staff hours value calculated from the function point count and other variables has been found to be in variance from the actual reported time. This metric plays a large role in various other metrics as well as the model fault prediction. The cause of the variation is possibly caused by the staff's strengths, weaknesses, points of emphasis, interest and many other human factors. To illustrate this point, a staff or project team may emphasize the gathering of requirements over the testing phase or a team may decide to spend most of the project performing coding and unit test tasks and perform little reviewing.

Related to the parameter optimization conclusion previously stated is the realization that continued research into metrics is necessary. As in this research, Agile development and other software development approaches will be adapted and metrics will be needed to apply the model to these development processes. Currently, there are multiple sources of data from numerous contributors, Jones, Kan, etc. and these sources need to be researched for more applicable metrics. Close inspection of the age, technology and practices used to gather all metrics is needed.

A primary conclusion of this study was the ability to extend the model to another lifecycle. This was accomplished through the function point count extension, discussed previously, and the mappings of phases in the new lifecycle to phases in which metrics are available. To accomplish this understanding of the underlying function provided in the new lifecycle was necessary. For instance, the feature list development phase, in the FDD lifecycle, was analyzed to be a requirements

gathering phase and metrics reflect that. Other phases, such as system modeling, required the knowledge that the architecture was being developed in that phase, thus, the phase was mapped to be a design phase with architecture development. Also important is the identifying a phase as a Fault Introduction or Fault Removal phase.

Finally, the optimization of SLI and the ability to perform control over the model provides many insights. On this small sample size of two small projects, through optimizing the SLI the model predictions were able to match the observed data. Another observation was the fact that in most cases the optimized SLI correlated to the expert opinion SLI.

6.2 Further Research

In the course of performing this research there arose various extensions to the study which can be further explored.

6.2.1 Extending the SLI Optimization to Preceding Phases

The SLI optimization of Fault Introduction – Fault Removal phases produced meaningful results. In the phases in which the optimization was performed the predicted and actual fault data correlated well, the SLIs produced by optimization aligned with the expert opinion SLI values and the model was able to be controlled.

This leads to the assumption that optimizing further back in the chain of phases would produce more clarity in the results. This is especially needed in the testing phases where many Fault Removal phases are encountered without a Fault Introduction phase preceding it. This makes it difficult to optimize both phases. For example in these two lifecycles, of the research, the ST phase is after a CI, UT and

IgT phase. Thus, to optimize ST to the nearest Fault Introduction phase, CO, would require optimizing through numerous other removal phases.

The benefits would seem to encourage further work in providing further analysis into the model equations to perform this task.

Overall, further work in the optimizing of model parameters would be beneficial. Striving for better model accuracy is the main goal of this further research and is related to performing Bayesian research to improve the quality of the parameters used by the model.

6.2.2 Extending the Model to Other Lifecycles

It was shown that the model may extend to another lifecycle. Work needs to be done to show that other lifecycles have phases that map to known phases which have available metrics, the ability to measure the effort, in terms of function point counts, of those phases and have fault introduction and fault removal phases.

6.2.3 Hidden Markov Model to Optimize Parameters

A Hidden Markov Model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with an unobserved state. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In an HMM, the state is not directly visible, but output dependent on the state that is visible [24]. Note that the adjective "hidden" refers to the state sequence through which the model passes, not to the parameters of the model. Even if the model parameters are known exactly, the model is still "hidden".

The lifecycles in this research can be represented by an HMM. The goal of research into the area of HMM is to determine if better optimization of the parameters is possible using HMM parameter optimization techniques.

6.2.4 Bayesian Analysis of Model Parameters

The metrics used in this study came from industry standards. These metrics are categorized by project type and size, in general. It would be advantageous to tailor these metrics, through repeated analysis, to the environment in which a practitioner is in. In this case, the accuracy of the model would be improved.

6.2.5 Phase Time

The fault prediction model for this research model used actual hours recorded and observed from the project team as the phase duration (in staff hours). The alternate is to use the estimate of staff hours given by industry data. Research into the effects of choosing one approach as opposed to the other may give improvements in the prediction of faults. The two approaches for assessing the phase time, in the two projects of this research, proved to have a varying degree of correlation.

Using the estimate of staff hours given by industry data also could improve the look ahead prediction capabilities by correcting for duration of phases that lie ahead.

Appendix A. Software Metric Data

Table 44 Patterns of Defect Prevention and Defect Removal Activities

	End-User	Web	MIS	Outsource	Commercial	Systems	Military
Pretest Removal							
Desk checking	15.00%	15.00%	15.00%	15.00%	15.00%	15.00%	15.00%
RE revision				30.00%	25.00%	20.00%	20.00%
DE review				40.00%	45.00%	45.00%	30.00%
Document review					20.00%	20.00%	20.00%
	End-User	Web	MIS	Outsource	Commercial	Systems	Military
Code Inspections					50.00%	60.00%	40.00%
Testing Activities							
Unit test	30.00%	30.00%	25.00%	25.00%	25.00%	25.00%	25.00%
Regression test				20.00%	20.00%	20.00%	20.00%
Integration test			30.00%	30.00%	30.00%	30.00%	30.00%
Perform test					15.00%	15.00%	20.00%
Systems test			35.00%	35.00%	35.00%	40.00%	35.00%
Accept. Test		25.00%		25.00%		25.00%	30.00%
Subtotal	30.00%	52.50%	76.11%	80.89%	91.88%	92.58%	93.63%
Cumul. Efficiency	52.40%	75.01%	88.63%	96.18%	99.32%	99.58%	99.33%
Number of Activities	3	4	7	11	14	16	18

Table 45 Defect Removal Efficiencies by Defect Origin

	Defects by Origin (%)			
Removal Step	**Requirements**	**Design**	**Coding**	**Documentation**
Prototyping	40	35	35	15
Requirements Review	40	15	0	5
Design Review	15	55	0	15
Code Inspection	20	40	65	25
Unit testing	1	5	20	0
System testing	10	15	35	20

Appendix B. Waterfall Project SLI from Expert Opinion

B.1 RQ SLI from Expertise

Table 46 RQ SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.3	0.061111111	experience (novice / expert)
20	0.037037037	0.2	0.007407407	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	0.2	0.018518519	schedule pressure (heavy / light)
30	0.055555556	0.2	0.011111111	use of methods/notations (crude / refined)
90	0.166666667	0.2	0.033333333	communications (poor / good)
10	0.018518519	0.3	0.005555556	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.1	0.007407407	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.212962963**	**Total Value**

B.2 RR SLI from Expertise

Table 47 RR SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.8	0.162962963	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.5	0.074074074	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.6	0.033333333	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.95	0.017592593	team relationships (poor / good)
20	0.037037037	0.35	0.012962963	management style (intrusive / supportive)
40	0.074074074	0.75	0.055555556	process integration method (crude / refined)
90	0.166666667	0.2	0.033333333	requirements volatility (chaotic / stable)
540	**1**		**0.662037037**	**Total Value**

B.3 DE SLI from Expertise

Table 48 DE SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.7	0.142592593	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.5	0.074074074	software complexity (difficult / simple)
50	0.092592593	0.2	0.018518519	schedule pressure (heavy / light)
30	0.055555556	0.8	0.044444444	use of methods/notations (crude / refined)
90	0.166666667	0.2	0.033333333	communications (poor / good)
10	0.018518519	0.5	0.009259259	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.5	0.037037037	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.412962963**	**Total Value**

B.4 DR SLI from Expertise

Table 49 DR SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.5	0.101851852	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.5	0.074074074	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.7	0.116666667	communications (poor / good)
10	0.018518519	0.85	0.015740741	team relationships (poor / good)
20	0.037037037	0.35	0.012962963	management style (intrusive / supportive)
40	0.074074074	0.5	0.037037037	process integration method (crude / refined)
90	0.166666667	0.2	0.033333333	requirements volatility (chaotic / stable)
540	**1**		**0.536111111**	**Total Value**

B.5 CO SLI from Expertise

Table 50 CO SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.5	0.101851852	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.5	0.074074074	software complexity (difficult / simple)
50	0.092592593	0.2	0.018518519	schedule pressure (heavy / light)
30	0.055555556	0.5	0.027777778	use of methods/notations (crude / refined)
90	0.166666667	0.4	0.066666667	communications (poor / good)
10	0.018518519	0.6	0.011111111	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.4	0.02962963	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.383333333**	**Total Value**

B.6 ST SLI from Expertise

Table 51 ST SLI Calculation From Expertise (Waterfall)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.8	0.162962963	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.5	0.074074074	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.8	0.044444444	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.85	0.015740741	team relationships (poor / good)
20	0.037037037	0.35	0.012962963	management style (intrusive / supportive)
40	0.074074074	0.8	0.059259259	process integration method (crude / refined)
90	0.166666667	0.2	0.033333333	requirements volatility (chaotic / stable)
540	**1**		**0.675**	**Total Value**

Appendix C. FDD Project SLI from Expert Opinion

C.1 RQ SLI from Expertise

Table 52 RQ SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.4	0.081481481	experience (novice / expert)
20	0.037037037	0.7	0.025925926	capability (low / high)
80	0.148148148	0.4	0.059259259	software complexity (difficult / simple)
50	0.092592593	0.9	0.083333333	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.6	0.1	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.425925926**	**Total Value**

C.2 RR SLI from Expertise

Table 53 RR SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.2	0.040740741	experience (novice/expert)
20	0.037037037	0.7	0.025925926	capability (low / high)
80	0.148148148	0.4	0.059259259	software complexity (difficult / simple)
50	0.092592593	0.9	0.083333333	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.6	0.1	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.385185185**	**Total Value**

C.3 SD SLI from Expertise

Table 54 SD SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.8	0.162962963	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.7	0.103703704	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.8	0.044444444	use of methods/notations (crude / refined)
90	0.166666667	0.8	0.133333333	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.8	0.059259259	process integration method (crude / refined)
90	0.166666667	0.3	0.05	requirements volatility (chaotic / stable)
540	**1**		**0.709259259**	**Total Value**

C.4 SDR SLI from Expertise

Table 55 SDR SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.3	0.061111111	experience (novice / expert)
20	0.037037037	0.7	0.025925926	capability (low / high)
80	0.148148148	0.4	0.059259259	software complexity (difficult / simple)
50	0.092592593	0.9	0.083333333	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.5	0.083333333	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.388888889**	**Total Value**

C.5 SM SLI from Expertise

Table 56 SM SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.7	0.142592593	experience (novice / expert)
20	0.037037037	0.8	0.02962963	capability (low / high)
80	0.148148148	0.7	0.103703704	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.6	0.033333333	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.6	0.044444444	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.646296296**	**Total Value**

C.6 SMR SLI from Expertise

Table 57 SMR SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factors
110	0.203703704	0.1	0.02037037	experience (novice / expert)
20	0.037037037	0.4	0.014814815	capability (low / high)
80	0.148148148	0.2	0.02962963	software complexity (difficult / simple)
50	0.092592593	0.9	0.083333333	schedule pressure (heavy / light)
30	0.055555556	0.2	0.011111111	use of methods/notations (crude / refined)
90	0.166666667	0.6	0.1	communications (poor / good)
10	0.018518519	0.6	0.011111111	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.309259259**	**Total Value**

C.7 FL SLI from Expertise

Table 58 FL SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.2	0.040740741	experience (novice / expert)
20	0.037037037	0.6	0.022222222	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.3	0.016666667	use of methods/notations (crude / refined)
90	0.166666667	0.6	0.1	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.3	0.022222222	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.388888889**	**Total Value**

C.8 FLR SLI from Expertise

Table 59 FLR SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.2	0.040740741	experience (novice / expert)
20	0.037037037	0.5	0.018518519	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.3	0.016666667	use of methods/notations (crude / refined)
90	0.166666667	0.5	0.083333333	communications (poor / good)
10	0.018518519	0.5	0.009259259	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.3	0.022222222	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.362962963**	**Total Value**

C.9 FLP SLI from Expertise

Table 60 FLP SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.2	0.040740741	experience (novice / expert)
20	0.037037037	0.7	0.025925926	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	0.9	0.083333333	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.6	0.1	communications (poor / good)
10	0.018518519	0.8	0.014814815	team relationships (poor / good)
20	0.037037037	0.2	0.007407407	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.37037037**	**Total Value**

C.10 DE(Iteration 0, 1, 2, 3) SLI from Expertise

Table 61 DE (0, 1, 2, 3) SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.8	0.162962963	experience (novice / expert)
20	0.037037037	0.9	0.033333333	capability (low / high)
80	0.148148148	0.6	0.088888889	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.7	0.038888889	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.9	0.016666667	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.8	0.059259259	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.677777778**	**Total Value**

C.11 DR(Iteration 0, 1, 2, 3) SLI from Expertise

Table 62 DR (0, 1, 2, 3) SLI Calculation from Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.2	0.040740741	experience (novice / expert)
20	0.037037037	0.4	0.014814815	capability (low / high)
80	0.148148148	0.4	0.059259259	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.3	0.05	communications (poor / good)
10	0.018518519	0.5	0.009259259	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.338888889**	**Total Value**

C.12 CO(Iteration 0, 1, 2, 3) SLI from Expertise

Table 63 CO (0, 1, 2, 3) SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.9	0.183333333	experience (novice / expert)
20	0.037037037	0.9	0.033333333	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.8	0.044444444	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.9	0.016666667	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.8	0.059259259	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.659259259**	**Total Value**

C.13 CI(Iteration 0, 1, 2, 3) SLI from Expertise

Table 64 CI (0, 1, 2, 3) SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.4	0.081481481	experience (novice / expert)
20	0.037037037	0.4	0.014814815	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.4	0.022222222	use of methods/notations (crude / refined)
90	0.166666667	0.5	0.083333333	communications (poor / good)
10	0.018518519	0.5	0.009259259	team relationships (poor / good)
20	0.037037037	0.5	0.018518519	management style (intrusive / supportive)
40	0.074074074	0.2	0.014814815	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.398148148**	**Total Value**

C.14 ST(Iteration 0, 1, 2, 3) SLI from Expertise

Table 65 ST (0, 1, 2, 3) SLI Calculation From Expertise (FDD)

Rank	Normalized Rank	Assessed Quality	Product	Influencing Factor
110	0.203703704	0.9	0.183333333	experience (novice / expert)
20	0.037037037	0.9	0.033333333	capability (low / high)
80	0.148148148	0.3	0.044444444	software complexity (difficult / simple)
50	0.092592593	1	0.092592593	schedule pressure (heavy / light)
30	0.055555556	0.8	0.044444444	use of methods/notations (crude / refined)
90	0.166666667	0.9	0.15	communications (poor / good)
10	0.018518519	0.9	0.016666667	team relationships (poor / good)
20	0.037037037	0.8	0.02962963	management style (intrusive / supportive)
40	0.074074074	0.6	0.044444444	process integration method (crude / refined)
90	0.166666667	0.1	0.016666667	requirements volatility (chaotic / stable)
540	**1**		**0.655555556**	**Total Value**

Appendix D. Test Case Parameters

D.1 Waterfall Project

D.1.1 Test Case #40

Time of Phase Start & End

Actual Time

RQ tStart:	0
RQ tEnd:	80.00
RR tStart:	80.00
RR tEnd:	128.00
DE tStart:	128.00
DE tEnd:	285.50
DR tStart:	285.50
DR tEnd:	353.00
CO tStart:	353.00
CO tEnd:	635.60
CI tStart:	635.60
CI tEnd:	651.60
UT tStart:	651.60
UT tEnd:	745.80
IgT tStart:	745.80
IgT tEnd:	840.00
ST tStart:	840.00
ST tEnd:	890.00
AT tStart:	890.00
AT tEnd:	934.00

Time per FP (t_{FP})

RQ:	1.00
RR:	0.74
DE:	3.39
DR:	0.74
CO:	3.34
CI:	0.95
UT:	0.96
IgT:	0.85
ST:	0.70
AT:	0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP): 3.17 (Weighted Mean)
Fault Fraction (fd): 0.10
Time per FP (t_{FP}): 1.00
F: 5.13
SLI: 0.213

DE:

Defect Potential(DP): 3.17 (Weighted Mean)
Fault Fraction (fd): 0.25
Time per FP (t_{FP}): 3.39
F: 5.67
SLI: 0.413

CO:

Defect Potential(DP): 3.17 (Weighted Mean)
Fault Fraction (fd): 0.40
Time per FP (t_{FP}): 3.34
F: 5.76
SLI: 0.383

Number of Function Points (N_{FP}): 59.28

Fault Removal Phases [$Z_a(t)$]

RQ Faults (Defect Removal Efficiency (DRE %))

DRE RQ: 15
DRE RR: 40
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20
DRE UT: 1
DRE IgT: 12.3
DRE ST: 10
DRE AT: 26.25

DE Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE DE: 15
DRE DR: 55
DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE ST: 15
DRE AT: 26.25

CO Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0

DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE ST: 35
DRE AT: 26.25

Aggregate DRE
None

Actual Faults
RR: 6
DR: 7
ST: 7

D.1.2 Test Case #41

Time of Phase Start & End
Actual Time
RQ tStart: 0
RQ tEnd: 80.00
RR tStart: 80.00
RR tEnd: 128.00
DE tStart: 128.00
DE tEnd: 285.50
DR tStart: 285.50
DR tEnd: 353.00
CO tStart: 353.00
CO tEnd: 635.60
CI tStart: 635.60
CI tEnd: 651.60
UT tStart: 651.60
UT tEnd: 745.80
IgT tStart: 745.80
IgT tEnd: 840.00
ST tStart: 840.00
ST tEnd: 890.00
AT tStart: 890.00
AT tEnd: 934.00

Time per FP (t_{FP})
RQ: 1.00
RR: 0.74

DE: 3.39
DR: 0.74
CO: 3.34
CI: 0.95
UT: 0.96
IgT: 0.85
ST: 0.70
AT: 0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.10	
Time per FP (t_{FP}):	1.00	
F:	5.13	
SLI:	0.213	

DE:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.25	
Time per FP (t_{FP}):	3.39	
F:	5.67	
SLI:	0.413	

CO:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.40	
Time per FP (t_{FP}):	3.34	
F:	5.76	
SLI:	0.383	

Number of Function Points (N_{FP}): 59.28

Fault Removal Phases [$Z_a(t)$]

RQ Faults (Defect Removal Efficiency (DRE %))

DRE RQ: 15
DRE RR: 40
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20
DRE UT: 1
DRE IgT: 12.3
DRE ST: 10
DRE AT: 26.25

DE Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0

DRE DE: 15
DRE DR: 55
DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE ST: 15
DRE AT: 26.25

CO Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE ST: 35
DRE AT: 26.25

Aggregate DRE

RR:

Max: 50 Min: 20 Modal: 30
SLI: 0.6623

DR:

Max: 65 Min: 30 Modal: 45
SLI: 0.5361

ST:

Max: 65 Min: 25 Modal: 50
SLI: 0.675

Actual Faults

RR: 6
DR: 7
ST: 7

D.1.3 Test Case #39

Time of Phase Start & End

Actual Time

RQ tStart: 0
RQ tEnd: 80.00
RR tStart: 80.00
RR tEnd: 128.00

DE tStart:	128.00
DE tEnd:	285.50
DR tStart:	285.50
DR tEnd:	353.00
CO tStart:	353.00
CO tEnd:	635.60
CI tStart:	635.60
CI tEnd:	651.60
UT tStart:	651.60
UT tEnd:	745.80
IgT tStart:	745.80
IgT tEnd:	840.00
ST tStart:	840.00
ST tEnd:	890.00
AT tStart:	890.00
AT tEnd:	934.00

Time per FP (t_{FP})

RQ:	1.00
RR:	0.74
DE:	3.39
DR:	0.74
CO:	3.34
CI:	0.95
UT:	0.96
IgT:	0.85
ST:	0.70
AT:	0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.10	
Time per FP (t_{FP}):	1.00	
F:	5.13	
SLI:	0.21	

DE:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.25	
Time per FP (t_{FP}):	3.39	
F:	5.67	
SLI:	0.43	

CO:

Defect Potential(DP):	3.17	(Weighted Mean)
Fault Fraction (fd):	0.40	
Time per FP (t_{FP}):	3.34	

F: 5.76
SLI: 0.383

Number of Function Points (N_{FP}): 59.28

Fault Removal Phases [$Z_a(t)$]

RQ Faults (Defect Removal Efficiency (DRE %))

DRE RQ: 15
DRE RR: 40
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20
DRE UT: 1
DRE IgT: 12.3
DRE ST: 10
DRE AT: 26.25

DE Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE DE: 15
DRE DR: 55
DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE ST: 15
DRE AT: 26.25

CO Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE ST: 35
DRE AT: 26.25

Aggregate DRE

RR:

Max: 50 Min: 20 Modal: 30
SLI: 0.65

DR:

Max: 65 Min: 30 Modal: 45

SLI: 0.53

ST:

Max: 65 Min: 25 Modal: 50
SLI: 0.675

Actual Faults

RR: 6
DR: 7
ST: 7

D.2 FDD Project

D.2.1 Test Case #27

Time of Phase Start & End

Actual Time

Time per FP (t_{FP})

RQ: 1.00
RR: 0.74
SD: 1.00
SDR: 0.74
SM: 0.56
SMR: 0.74
FL: 1.00
FLR: 0.74
FLP: 0.30
DE: 1.54
DR: 0.74
CO: 3.34
CI: 0.95
UT: 0.96
IgT: 0.85
ST: 0.70
AT: 0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP): 3.219
Fault Fraction (fd): 0.10
Time per FP (t_{FP}): 1.00
F: 5.13
SLI: 0.4259

SD:

Defect Potential(DP): 3.219
Fault Fraction (fd): 0.25
Time per FP (t_{FP}): 1.00
F: 4.34
SLI: 0.7093

SM:

Defect Potential(DP): 3.219
Fault Fraction (fd): 0.25
Time per FP (t_{FP}): 0.56
F: 5.39
SLI: 0.6463

FL:

Defect Potential(DP): 3.219
Fault Fraction (fd): 0.10
Time per FP (t_{FP}): 1.00
F: 5.13
SLI: 0.3889

FLP:

Defect Potential(DP): 3.219
Fault Fraction (fd): 0.25
Time per FP (t_{FP}): 0.30
F: 5.18
SLI: 0.3704

DE:

DE 0 Defect Potential(DP): 2.912
DE 1 Defect Potential(DP): 2.648
DE 2 Defect Potential(DP): 2.719
DE 3 Defect Potential(DP): 2.857
Fault Fraction (fd): 0.25
Time per FP (t_{FP}): 1.54
F: 6.74
SLI0: 0.6778
SLI1: 0.6778
SLI2: 0.6778
SLI3: 0.6778

CO:

CO 0 Defect Potential(DP): 2.912
CO 1 Defect Potential(DP): 2.648
CO 2 Defect Potential(DP): 2.719
CO 3 Defect Potential(DP): 2.857
Fault Fraction (fd): 0.40
Time per FP (t_{FP}): 3.34
F: 5.77
SLI0: 0.6593

SLI1: 0.6593
SLI2: 0.6593
SLI3: 0.6593

Number of Function Points (N_{FP}):
N_{FP}(Total): 147.34
N_{FP}(It. 0): 66.73
N_{FP}(It. 0 RT): 66.73
N_{FP}(It. 1): 19.77
N_{FP}(It. 1 RT): 86.50
N_{FP}(It. 2): 27.45
N_{FP}(It. 2 RT): 114.48
N_{FP}(It. 3): 51.73
N_{FP}(It. 3 RT): 159.00

Fault Removal Phases [$Z_a(t)$]
RQ Faults (Defect Removal Efficiency (DRE %))
DRE RQ: 15
DRE RR: 40
DRE SD: 15
DRE SDR: 15
DRE SM: 15
DRE SMR: 15
DRE FL: 15
DRE FLR: 40
DRE FLP: 15
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20
DRE UT: 1
DRE IgT: 12.3
DRE AT: 26.25
DRE ST: 10
DE Faults (Defect Removal Efficiency (DRE))
DRE RQ: 0
DRE RR: 0
DRE SD: 15
DRE SDR: 55
DRE SM: 15
DRE SMR: 55
DRE FL: 15
DRE FLR: 15
DRE FLP: 15
DRE DE: 15
DRE DR: 55

DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE AT: 26.25
DRE ST: 15

CO Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE SD: 0
DRE SDR: 0
DRE SM: 0
DRE SMR: 0
DRE FL: 0
DRE FLR: 0
DRE FLP: 0
DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE AT: 26.25
DRE ST: 35

Aggregate DRE

None

D.2.2 Test Case #28

Time of Phase Start & End

Actual Time

Time per FP (t_{FP})

RQ: 1.00
RR: 0.74
SD: 1.00
SDR: 0.74
SM: 0.56
SMR: 0.74
FL: 1.00
FLR: 0.74
FLP: 0.30
DE: 1.54

DR: 0.74
CO: 3.34
CI: 0.95
UT: 0.96
IgT: 0.85
ST: 0.70
AT: 0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.10
Time per FP (t_{FP}):	1.00
F:	5.13
SLI:	0.4259

SD:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	1.00
F:	4.34
SLI:	0.7093

SM:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	0.56
F:	5.39
SLI:	0.6463

FL:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.10
Time per FP (t_{FP}):	1.00
F:	5.13
SLI:	0.3889

FLP:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	0.30
F:	5.18
SLI:	0.3704

DE:

DE 0 Defect Potential(DP):	2.912
DE 1 Defect Potential(DP):	2.648
DE 2 Defect Potential(DP):	2.719
DE 3 Defect Potential(DP):	2.857
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	1.54

F: 6.74
SLI0: 0.6778
SLI1: 0.6778
SLI2: 0.6778
SLI3: 0.6778

CO:

CO 0 Defect Potential(DP): 2.912
CO 1 Defect Potential(DP): 2.648
CO 2 Defect Potential(DP): 2.719
CO 3 Defect Potential(DP): 2.857
Fault Fraction (fd): 0.40
Time per FP (t_{FP}): 3.34
F: 5.77
SLI0: 0.6593
SLI1: 0.6593
SLI2: 0.6593
SLI3: 0.6593

Number of Function Points (N_{FP}):

N_{FP}(Total): 147.34
N_{FP}(It. 0): 66.73
N_{FP}(It. 0 RT): 66.73
N_{FP}(It. 1): 19.77
N_{FP}(It. 1 RT): 86.50
N_{FP}(It. 2): 27.45
N_{FP}(It. 2 RT): 114.48
N_{FP}(It. 3): 51.73
N_{FP}(It. 3 RT): 159.00

Fault Removal Phases [$Z_a(t)$]

RQ Faults (Defect Removal Efficiency (DRE %))

DRE RQ: 15
DRE RR: 40
DRE SD: 15
DRE SDR: 15
DRE SM: 15
DRE SMR: 15
DRE FL: 15
DRE FLR: 40
DRE FLP: 15
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20

DRE UT: 1
DRE IgT: 12.3
DRE AT: 26.25
DRE ST: 10

DE Faults (Defect Removal Efficiency (DRE))
DRE RQ: 0
DRE RR: 0
DRE SD: 15
DRE SDR: 55
DRE SM: 15
DRE SMR: 55
DRE FL: 15
DRE FLR: 15
DRE FLP: 15
DRE DE: 15
DRE DR: 55
DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE AT: 26.25
DRE ST: 15

CO Faults (Defect Removal Efficiency (DRE))
DRE RQ: 0
DRE RR: 0
DRE SD: 0
DRE SDR: 0
DRE SM: 0
DRE SMR: 0
DRE FL: 0
DRE FLR: 0
DRE FLP: 0
DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE AT: 26.25
DRE ST: 35

Aggregate DRE
RR:
Max: 50 Min: 20 Modal: 30
SLI: 0.3852
SDR:

Max: 60 Min: 30 Modal: 40
SLI: 0.3889

SMR:
Max: 65 Min: 30 Modal: 45
SLI: 0.3093

FLR:
Max: 50 Min: 20 Modal: 30
SLI: 0.363

DR0:
Max: 65 Min: 30 Modal: 45
SLI: 0.3389

ST0:
Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR1:
Max: 65 Min: 30 Modal: 45
SLI: 0.3389

ST1:
Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR2:
Max: 65 Min: 30 Modal: 45
SLI: 0.3389

ST2:
Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR3:
Max: 65 Min: 30 Modal: 45
SLI: 0.3389

CI3:
Max: 85 Min: 35 Modal: 60
SLI: 0.3981

ST3:
Max: 65 Min: 25 Modal: 50
SLI: 0.6556

D.2.3 Test Case #29

Time of Phase Start & End
Actual Time

Time per FP (t_{FP})
RQ: 1.00
RR: 0.74
SD: 1.00

SDR: 0.74
SM: 0.56
SMR: 0.74
FL: 1.00
FLR: 0.74
FLP: 0.30
DE: 1.54
DR: 0.74
CO: 3.34
CI: 0.95
UT: 0.96
IgT: 0.85
ST: 0.70
AT: 0.58

Fault Introduction Phases [$\nu(t)$ $\mu_H(t)$]

RQ:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.10
Time per FP (t_{FP}):	1.00
F:	5.13
SLI:	0.56

SD:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	1.00
F:	4.34
SLI:	0.77

SM:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	0.56
F:	5.39
SLI:	0.64

FL:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.10
Time per FP (t_{FP}):	1.00
F:	5.13
SLI:	0.21

FLP:

Defect Potential(DP):	3.219
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	0.30
F:	5.18
SLI:	0.3704

DE:

DE 0 Defect Potential(DP):	2.912
DE 1 Defect Potential(DP):	2.648
DE 2 Defect Potential(DP):	2.719
DE 3 Defect Potential(DP):	2.857
Fault Fraction (fd):	0.25
Time per FP (t_{FP}):	1.54
F:	6.74
SLI0:	0.71
SLI1:	0.59
SLI2:	0.95
SLI3:	0.68

CO:

CO 0 Defect Potential(DP):	2.912
CO 1 Defect Potential(DP):	2.648
CO 2 Defect Potential(DP):	2.719
CO 3 Defect Potential(DP):	2.857
Fault Fraction (fd):	0.40
Time per FP (t_{FP}):	3.34
F:	5.77
SLI0:	0.6593
SLI1:	0.6593
SLI2:	0.6593
SLI3:	0.71

Number of Function Points (N_{FP}):

N_{FP}(Total):	147.34
N_{FP}(It. 0):	66.73
N_{FP}(It. 0 RT):	66.73
N_{FP}(It. 1):	19.77
N_{FP}(It. 1 RT):	86.50
N_{FP}(It. 2):	27.45
N_{FP}(It. 2 RT):	114.48
N_{FP}(It. 3):	51.73
N_{FP}(It. 3 RT):	159.00

Fault Removal Phases [$Z_a(t)$]

RQ Faults (Defect Removal Efficiency (DRE %))

DRE RQ:	15
DRE RR:	40
DRE SD:	15
DRE SDR:	15
DRE SM:	15
DRE SMR:	15

DRE FL: 15
DRE FLR: 40
DRE FLP: 15
DRE DE: 15
DRE DR: 15
DRE CO: 15
DRE CI: 20
DRE UT: 1
DRE IgT: 12.3
DRE AT: 26.25
DRE ST: 10

DE Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE SD: 15
DRE SDR: 55
DRE SM: 15
DRE SMR: 55
DRE FL: 15
DRE FLR: 15
DRE FLP: 15
DRE DE: 15
DRE DR: 55
DRE CO: 15
DRE CI: 40
DRE UT: 5
DRE IgT: 35.7
DRE AT: 26.25
DRE ST: 15

CO Faults (Defect Removal Efficiency (DRE))

DRE RQ: 0
DRE RR: 0
DRE SD: 0
DRE SDR: 0
DRE SM: 0
DRE SMR: 0
DRE FL: 0
DRE FLR: 0
DRE FLP: 0
DRE DE: 0
DRE DR: 0
DRE CO: 15
DRE CI: 65
DRE UT: 20
DRE IgT: 62.1
DRE AT: 26.25

DRE ST: 35

Aggregate DRE

RR:

Max: 50 Min: 20 Modal: 30
SLI: 0.41

SDR:

Max: 60 Min: 30 Modal: 40
SLI: 0.28

SMR:

Max: 65 Min: 30 Modal: 45
SLI: 0.33

FLR:

Max: 50 Min: 20 Modal: 30
SLI: 0.54

DR0:

Max: 65 Min: 30 Modal: 45
SLI: 0.12

ST0:

Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR1:

Max: 65 Min: 30 Modal: 45
SLI: 0.28

ST1:

Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR2:

Max: 65 Min: 30 Modal: 45
SLI: 0.28

ST2:

Max: 65 Min: 25 Modal: 50
SLI: 0.6556

DR3:

Max: 65 Min: 30 Modal: 45
SLI: 0.41

CI3:

Max: 85 Min: 35 Modal: 60
SLI: 0.26

ST3:

Max: 65 Min: 25 Modal: 50
SLI: 0.6556

Bibliography

[1] J. D. Musa, A. Iannino and K. Okumoto, *Software Reliability: Measurement,Prediction, Application*, New York, USA: McGraw-Hill, 1987.

[2] A. M. Neufelder, *Ensuring Software Reliability*, New York, USA: Marcel Dekker, 1993.

[3] M. Xie, *Software Reliability Modeling*, Singapore: World Scientific, 1991.

[4] K. Beck, et al., "Manifesto for Agile Software Development," http://agilemanifesto.org/ [accessed May 7, 2011].

[5] S. R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[6] C. Jones, *Applied Software Measurement: Assuring Productivity and Quality,* 2nd ed. New York, NY, USA: McGraw-Hill, 1996.

[7] C. Jones, *Applied Software Measurement: Global Analysis of Productivity and Quality,* 3rd ed. New York, NY, USA: McGraw-Hill, 2008.

[8] S. H. Kan, *Metrics and Models in Software Quality Engineering,* 2nd ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2003.

[9] D. Garmus and D. Herron, *Function Point Analysis: Measurement Practices for Successful Software Projects,* Boston, MA, USA: Addison-Wesley, 2003.

[10] C. Jones, *Estimating Software Costs: Bringing Realism to Estimating*, 2nd ed. New York, NY, USA: McGraw-Hill, 2007.

[11] C. Jones, *Software Assessments, Benchmarks, and Best Practices*, 2nd ed. New York, NY, USA: McGraw-Hill, 2000.

[12] M. A. Stutzke and C. S. Smidts, "A Stochastic Model of Fault Introduction and Removal During Software Development," *IEEE Transactions on Reliability Engineering*, vol. 50, no. 2, June 2001.

[13] W. W. Royce, "Managing the Development of Large Software Systems," *Proceedings of IEEE WESCON*, 1-9, August 1970.

[14] C. Jones, *Software Engineering Best Practices*, 1st ed. New York, NY, USA: McGraw-Hill, 2010.

[15] C. Jones, "Software Quality in 2008: A Survey of the State of the Art," U.S. Averages for Software Quality, January 30, 2008, e-mail to author, June 15, 2009.

[16] C. Jones, "Software Quality in 2008: A Survey of the State of the Art," Poor Software Quality – Malpractice (Large Client-Sever Projects (> 5000 FP)), January 30, 2008, e-mail to author, June 15, 2009.

[17] C. Jones, "Software Quality in 2008: A Survey of the State of the Art," Best in Class Software Quality (Systems Software > SEI CMM Level 3), January 30, 2008, e-mail to author, June 15, 2009.

[18] A. Hoyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods,* 1st ed. New York, NY, USA: John Wiley & Sons, 1994.

[19] B. Kirwan, *A Guide to Practical Human Reliability Assessment*, 1st ed. Boca Raton, FL, USA: CRC Press, 1994.

[20] E. J. Henley and H. Kumamoto, *Reliability Engineering and Risk Assessment*, 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1981.

[21] C. Jones, “Overview of United States Software Industry Results Circa 2008”, Capers Jones & Associates LLC, 2008, e-mail to author, June 15, 2009.

[22] J. Verzani, *Using R for Introductory Statistics*, 1st ed. Boca Raton, FL, USA: CRC Press, 2005.

[23] C. S. Smidts, Y. Shi, M. Li, W. Kong and J. Dai, "A Large Scale Validation of a Methodology for Assessing Software Reliability," Nuclear Regulatory Commission NUREG/CR-7042, 2011.

[24] W. Zucchini and I. L. MacDonald, *Hidden Markov Models for Time Series An Introduction Using R*, 1st ed. Boca Raton, FL, USA: CRC Press, 2009.

[25] C. S. Smidts, B. Li, M. Li and Z. Li, “Software Reliability Models,” in *The Encyclopedia of Software Engineering,* edited by J. J. Marciniak, 1594-1610. New York: John Wiley & Sons, 2002.

[26] L. Williams, “Agile Software Development Methods and Practices,” Lecture, training session, Rochester, MN, April 25, 2008.

[27] C. S. Smidts and M. Li, “Preliminary Validation of a Methodology for Assessing Software Quality,” Nuclear Regulatory Commission NUREG/CR-6848, 2004.

[28] A. Cockburn, *Crystal Clear: A Human-Powered Methodologyfor Small Teams*, 1st ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2005.

[29] D. E. Embrey, “The Use of Performance Shaping Factors and Quantified Expert Judgement in the Evaluation of Human Reliability: An Initial Appraisal,” Nuclear Regulatory Commission NUREG/CR-2986, 1983.

[30] E. M. Dougherty and J.R Fragola, *Human Reliability Analysis: A Systems Engineering Approach with Nuclear Power Plant Applications*, New York: John Wiley & Sons, 1987.

[31] J. Reason, *Human Error*, Cambridge Univ. Press, 1990.

[32] C. Tichenor, e-mail to David Johnson, May 14, 2010.

[33] C. D. Schunn and D. Wallach, "Evaluating Goodness-of-Fit in Comparison of Models to Data," W. Tack (Ed.), Psychologie der Kognition: Reden and Vorträge anlässlich der Emeritierung von Werner Tack (pp. 115-154). Saarbrueken, Germany: University of Saarland Press (2005).